CCN-CERT

BP/07

Recommandations pour la mise en œuvre de HTTPS

RAPPORT DE BONNES PRATIQUES

JANVIER **2022**

ccn-cert centro criptológico nacional

CCN centro criptológico nacional

Editeur :

Paseo de la Castellana 109, 28046 Madrid

Date d'émission : janvier 2022

LIMITATION DE LA RESPONSABILITÉ

Le présent document est fourni conformément aux termes qu'il contient, rejetant expressément tout type de garantie implicite qui puisse y être liée. Le Centre National de Cryptologie ne peut en aucun cas être tenu responsable des dommages et préjudices —directs, indirects, fortuits ou extraordinaires— dérivés de l'utilisation de l'information et du logiciel contenus dans ce document, même s'il a été averti de cette possibilité[1].

AVIS JURIDIQUE

La reproduction totale ou partielle de ce document par quelque moyen ou procédé que ce soit —y compris la reprographie et le traitement informatique— et la location ou le prêt public de copies sont strictement interdites sans l'autorisation écrite du Centre National de Cryptologie, sous peine des sanctions prévues par la loi.

---

[1] Raúl Siles, fondateur et analyste de sécurité chez DinoSec, a participé à l'élaboration et à la modification de ce document et de ses annexes.

Index

Index

1. À propos du CCN-CERT

Le CCN-CERT (www.ccn-cert.cni.es) est la capacité de réponse aux incidents de sécurité de l'information du Centre National de Cryptologie, CCN (www.ccn.cni.es). Ce service a été créé en 2006 en tant que **CERT gouvernemental national** et ses fonctions sont définies dans la Loi 11/2002 réglementant le Centre National de Renseignement (CNI), le RD 421/2004 réglementant le CCN et le RD 3/2010, du 8 janvier, réglementant le Schéma National de sécurité (ENS), modifié par le RD 951/2015 du 23 octobre.

D'après la réglementation précédente, le CCN-CERT est l'organisme chargé de gérer les incidents de nature cyber qui affectent aussi bien les **systèmes informatiques du secteur public**, des **entreprises** et des **organisations représentant un intérêt stratégique** pour le pays, tout comme les systèmes classifiés. Sa mission est donc de contribuer à l'amélioration de la cybersécurité espagnole, en agissant comme centre national de veille, d'alerte et de réponse aux attaques informatiques. Le CERT assiste les organismes à mettre en place des moyens de protection nécessaires pour faire face aux menaces et à répondre aux attaques dont ils sont victimes d'une manière rapide et efficace.

Le CCN-CERT est la capacité de réponse aux incidents de sécurité de l'information du Centre National de Cryptologie

2. Sommaire exécutif

La prolifération des technologies Web au cours des dernières années, ainsi que l'augmentation de leurs capacités, fonctionnalités, caractéristiques et possibilités d'utilisation, permettant l'exécution de pratiquement n'importe quelle gestion ou tâche de la vie quotidienne, aussi bien personnelle que professionnelle, nous oblige à accorder une importance majeure à la sécurisation des communications sur internet par le biais du protocole HTTPS, capable de fournir à la fois des mécanismes d'authentification, de chiffrement et d'intégrité.

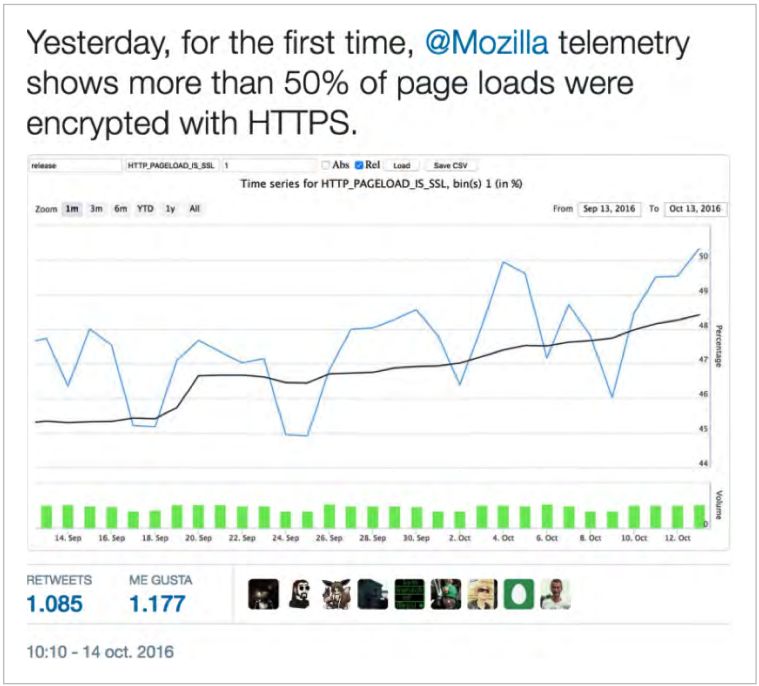
Il est donc nécessaire de promouvoir l'utilisation de technologies Web présentant un niveau de sécurité plus élevé grâce au protocole HTTPS, par opposition au protocole HTTP, aussi bien dans les environnements d'entreprise que pour un usage privé, et indépendamment du fait que l'on y accède à partir d'ordinateurs traditionnels ou de dispositifs mobiles, ou de tout autre dispositif technologique, comme ceux associés à l'Internet des objets (IoT, Internet of Things).

L'évolution du HTTPS dans l'industrie a connu un progrès très significatif au cours des dernières années, notamment en 2016 et 2017 [Réf. - 2], où pour la première fois dans l'histoire, les données de télémétrie recueillies par Mozilla montrent qu'en octobre 2016, le trafic Web HTTPS mondial a dépassé le trafic HTTP (plus de 50 %). Cette tendance à la hausse de l'utilisation de HTTPS par rapport à HTTP se poursuit tout au long de l'année 2017, pour atteindre près de 55 % en mars[2].

La prolifération des technologies Web au cours des dernières années nous oblige à accorder une importance majeure à la sécurisation des communications sur internet par le biais du protocole HTTPS

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

Figure 1.
Trafic Web mondial selon Mozilla : HTTPS vs. HTTP.
Source : Mozilla[3]



L'une des principales raisons pour lesquelles le protocole HTTPS devrait être utilisé pour l'accès aux environnements, serveurs et applications Web réside dans ses caractéristiques de sécurité supplémentaires, où l'utilisateur peut confirmer qu'il se connecte à l'environnement Web souhaité (authentification et identification). Un autre avantage est que tout le trafic échangé entre l'utilisateur et l'environnement Web est chiffré, ce qui garantit l'intégrité des données transmises et empêche le trafic d'être intercepté et manipulé par un tiers.

De nos jours, tout le trafic Web échangé via le protocole HTTP devrait être considéré comme sensible, même lorsqu'on utilise des serveurs Web publics et des applications où l'authentification n'est pas requise, car la confidentialité des utilisateurs et la protection des données échangées doivent être assurées à tout moment. Pour cette raison, il est nécessaire de généraliser l'utilisation du protocole HTTPS à tous les environnements Web, tant publics que privés (ou internes), quelle que soit l'organisation ou l'entreprise.

[3] https://twitter.com/0xjosh/status/786971412959420424, https://twitter.com/letsencrypt/status/786977436109934592

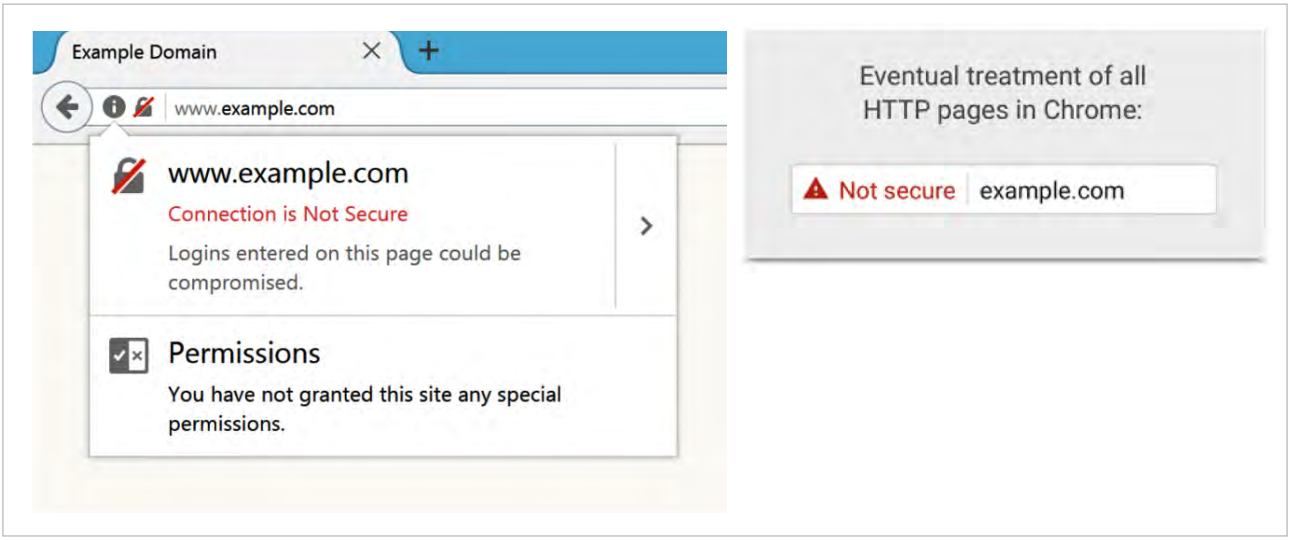


Figure 2.
Messages indiquant qu'un serveur Web HTTP n'est pas sécurisé. Source : Firefox et Chrome.

Un exemple de cette tendance, du point de vue des services Internet largement utilisés, s'est concrétisé en octobre 2011, lorsque Google a converti son navigateur Web au protocole HTTPS[4], un service accessible au public (où aucune authentification n'est requise) mais où les termes de recherche personnalisés utilisés par les utilisateurs doivent être protégés, garantissant ainsi leur confidentialité. D'autre part, du point de vue des clients Web, en janvier 2017, les principaux navigateurs Web (p. ex. Firefox 51[5] et Chrome 56[6]) ont commencé à mettre en évidence —de manière négative— les services Internet qui ne font pas usage du HTTPS et, en particulier, ceux qui requêtent des données sensibles à l'utilisateur, comme les identifiants ou les cartes de crédit, en signalant le site Web en rouge et accompagné du message "Not secure" (ou "Insecure").

Bien que les technologies Web soient couramment utilisées pour les communications personnelles et professionnelles, privées et pertinentes, pour l'échange de données sensibles et pour l'accès à de nombreux services, probablement en raison de leur utilisation répandue et de leur facilité d'emploi, le niveau de perception de la menace réelle pour la sécurité que représente la non-utilisation du protocole HTTPS n'est pas suffisamment élevé parmi les utilisateurs finaux et les organisations. Les environnements Web des organisations, et les communications HTTP des utilisateurs, sont souvent soumis à de nombreuses

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

attaques qui, de ce fait, permettent d'accéder à des informations sensibles, notamment des données personnelles, voire des identifiants de session et des informations d'identification qui permettraient d'usurper l'identité de leurs propriétaires.

Il convient de noter que le protocole HTTPS n'atténue que certaines attaques spécifiques associées aux technologies Web. Des mesures de sécurité supplémentaires doivent être mises en place pour protéger les serveurs et les applications Web contre d'autres attaques, telles que l'injection SQL (SQLi), XSS (*Cross-Site Scripting*), CSRF (*Cross-Site Request Forgery*), les attaques de gestion de session, LFI/RFI, etc.

La sensibilisation à l'usage exclusif du protocole HTTPS en entreprise et la transmission de bonnes pratiques aux utilisateurs pour sa mise en œuvre (conception, configuration et maintenance) dans les environnements Web constituent la meilleure défense pour prévenir et atténuer ce type particulier d'incidents et de menaces Web. L'objectif de ce document est de décrire ces pratiques afin d'aider les responsables des environnements Web à protéger à la fois leurs serveurs et leurs applications Web ainsi que les utilisateurs finaux, en approfondissant la configuration et l'utilisation des mécanismes de protection actuellement existants.

À cette fin, un ensemble détaillé de directives et de recommandations techniques en matière de sécurité sera fourni afin d'améliorer la sécurité et d'accroître les capacités de protection de la mise en œuvre du protocole HTTPS dans les environnements Web. En outre, la section "ANNEXE C. RÉFÉRENCES" fournit plusieurs références et une documentation permettant d'approfondir tous les aspects de la sécurité décrits dans ce rapport[7], en complément des nombreuses références figurant dans les notes de bas de page des différentes sections du rapport.

Ce rapport de bonnes pratiques, "Recommandations pour la mise en œuvre de HTTPS" (CCN-CERT BP-01/17), par sa nature plus technique, s'adresse à un public cible légèrement différent de celui des autres rapports de bonnes pratiques de CCN-CERT [Réf. - 1], dont le public cible est l'utilisateur final des technologies analysées. Dans ce cas, le public cible est constitué des administrateurs et/ou des responsables de la sécurité technique des environnements, des serveurs et des applications Web qui doivent évaluer et mettre en œuvre le support du protocole HTTPS de la manière la plus sûre possible.

La sensibilisation à l'usage exclusif du protocole HTTPS en entreprise et la transmission de bonnes pratiques aux utilisateurs pour sa mise en œuvre dans les environnements Web constituent la meilleure défense pour prévenir et atténuer ce type particulier d'incidents et de menaces Web

[7] Le livre "Bulletproof SSL and TLS" [Réf. - 10] est particulièrement intéressant.

3. Bonnes pratiques et recommandations pour la mise en œuvre de HTTPS

Ce rapport fournit des détails techniques pour la mise en œuvre et le déploiement sécurisés du protocole HTTPS, basé sur TLS (*Transport Layer Security*), dans les environnements, serveurs et applications Web. Afin de permettre au lecteur de mieux comprendre le raisonnement qui sous-tend les diverses recommandations de sécurité décrites ci-dessous, nous fournirons dans certains cas une brève description ou des références aux menaces et incidents de sécurité qui affectent aujourd'hui les implémentations HTTPS dans les environnements Web, ainsi que certaines des techniques les plus couramment utilisées par les attaquants.

L'ensemble des recommandations présentées est divisé en plusieurs sections, chacune d'entre elles étant liée à différents éléments, capacités et fonctionnalités associés aux implémentations HTTPS, notamment l'accès par les ports TCP/IP associés aux services Web, la génération de clés, la gestion des certificats numériques et d'autres considérations associées à l'infrastructure globale de clés publiques (PKI) de l'Internet, les protocoles SSL et TLS et leurs différentes versions, algorithmes et suites cryptographiques, la gestion des certificats numériques et d'autres considérations liées à l'infrastructure à clé pu-

blique (PKI) mondiale de l'internet, les protocoles SSL et TLS et leurs différentes versions, les algorithmes et suites cryptographiques, les mécanismes de révocation des certificats numériques, les recommandations pour la publication de contenu Web et le déploiement d'applications Web via HTTPS, etc. À la fin de chaque section se trouve un résumé des principales recommandations.

L'objectif de ces recommandations est que l'administrateur et/ou le responsable technique de la sécurité des environnements, serveurs et applications Web puisse augmenter le niveau de protection et de robustesse des services Web en fonction de la mise en œuvre et du support disponible pour le protocole HTTPS, tant du point de vue de sa conception et de sa configuration que de sa gestion et de sa maintenance quotidiennes, évitant ainsi que ces services Web ne soient victimes d'attaques connues sur HTTP et HTTPS. En outre, les implications possibles de la façon dont les mécanismes de sécurité proposés peuvent affecter les performances de communication de l'environnement Web seront prises en compte à tout moment et des recommandations seront fournies pour améliorer les performances du trafic Web même en utilisant HTTPS.

Il faut tenir compte du fait que certaines des caractéristiques décrites et, par conséquent, des recommandations de sécurité proposées, dépendent du type de système d'exploitation utilisé par l'environnement Web (Linux, BSD, Unix, Windows, etc.), du type de serveur Web utilisé (Apache, nginx, IIS, etc.), du type de serveur d'applications (Weblogic, Websphere, JBoss/WidFly, Tomcat, etc.), ainsi que des versions de tous ces éléments, et de l'existence d'autres composants périmétriques, tels que les équilibreurs de charge, les réseaux de diffusion de contenu (CDN-*Content Delivery Network*, reverse proxies, etc.), ainsi que les versions de tous ces éléments, et l'existence d'autres composants du périmètre, tels que les équilibreurs de charge, les réseaux de diffusion de contenu (CDN), les reverse proxies, les pare-feu, les pare-feu pour applications Web (WAF), les concentrateurs ou terminateurs HTTPS, etc. Par conséquent, toutes les recommandations données ne s'appliquent pas nécessairement à tous les composants de l'environnement Web. Dans tous les cas, il est recommandé d'appliquer autant de recommandations que possible.

Les références et les exemples de configuration technique plus spécifiques cités tout au long de ce rapport utilisent OpenSSL[8], en tant que bibliothèque (open source) potentiellement la plus utilisée pour mettre en œuvre les protocoles SSL et TLS, et Apache[9], en tant que serveur

L'objectif de ces recommandations est que l'administrateur et/ou le responsable technique de la sécurité des environnements, serveurs et applications Web puisse augmenter le niveau de protection et de robustesse des services Web

[8] https://www.openssl.org

[9] https://www.apache.org (https://httpd.apache.org)

Web (également open source) le plus utilisé sur les sites Web actifs sur Internet aujourd'hui et au cours des dernières années (avec près de 46 % de part de marché en février 2017[10]). En 2022, la situation a changé, Apache ayant chuté à 23,9 % et nginx ayant augmenté à 26,6 %, actuellement en tête en termes de parts de marché. Par conséquent, les administrateurs et/ou les responsables de la sécurité technique des environnements, serveurs et applications Web qui doivent évaluer et mettre en œuvre le support du protocole HTTPS doivent identifier en détail, dans chaque cas, les possibilités existantes et les particularités de configuration des technologies Web utilisées dans l'environnement où ce rapport sera appliqué.

Du point de vue de la sécurité, il est recommandé de mettre en œuvre simultanément et de manière complémentaire l'ensemble des mécanismes de sécurité décrits ci-dessus, en appliquant le principe de défense en profondeur. Il faut tenir compte du fait que tous les navigateurs et clients Web ne prennent pas en charge les différentes normes et mécanismes de sécurité proposés. Par conséquent, l'application de multiples mesures complémentaires, et bien qu'elles puissent parfois sembler répétitives (comme p. ex., l'utilisation d'une redirection 301 de HTTP à HTTPS, l'utilisation de l'en-tête HSTS ou l'utilisation de l'en-tête CSP pour mettre à jour les requêtes HTTP non sécurisées en requêtes HTTPS), permettra d'adapter l'environnement Web aux capacités disponibles chez les différents clients qui y accéderont.

Une fois que toutes ou la plupart des recommandations suggérées dans ce rapport ont été mises en œuvre, il est recommandé de vérifier l'état général de la mise en œuvre du protocole HTTPS pour les environnements, serveurs et applications Web publics à l'aide du service "SSL Server Test" de Qualys SSL Labs [Réf. - 4]. Lors de l'utilisation du service, il est recommandé de sélectionner l'option "*Ne pas afficher les résultats sur les tableaux*" avant d'effectuer l'analyse, afin de ne pas apparaître dans la liste publique des sites Web les plus récemment analysés, ou avec des résultats meilleurs ou moins bons[11]. L'analyse doit se concentrer sur l'obtention d'une note finale de A ou A+. Toute note inférieure doit être considérée comme une anomalie ou une faiblesse, et doit être traitée. Il convient de garder à l'esprit qu'une note de A aujourd'hui peut être associée à une note de B dans quelques mois, car les mécanismes de sécurité HTTPS et les exigences minimales associées aux bonnes pratiques évoluent, s'améliorent et sont révisés en permanence [Réf. - 21].

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

[11] Il convient de noter que rien n'empêche un tiers de fournir l'adresse (ou le nom d'hôte) du serveur Web local par le biais de ce service, sans sélectionner cette option, de sorte qu'il serait répertorié dans les différentes catégories mentionnées, accessibles au public.

3.1. Accès aux ports TCP/IP associés aux services Web

Les services Web sont associés par défaut au port TCP/80 dans le cas du protocole HTTP, et au port TCP/443 dans le cas du protocole HTTPS, qui fait en outre appel à des mécanismes d'authentification et de chiffrement utilisant TLS.

Les services Web sont associés par défaut au port TCP/80 dans le cas du protocole HTTP, et au port TCP/443 dans le cas du protocole HTTPS

Lors du déploiement d'un service Web, et en suivant la tendance industrielle axée sur la conversion et le basculement du Web uniquement sur le protocole HTTPS (éliminant autant que possible l'utilisation du protocole HTTP), il est recommandé d'utiliser exclusivement HTTPS, il est donc nécessaire d'avoir le port TCP/443 ouvert dans l'environnement Web.

Toutefois, les navigateurs et clients Web utilisent actuellement le protocole HTTP et le port TCP/80 par défaut lorsque le protocole à utiliser n'est pas spécifié ("http://" ou "https://"), par exemple lorsque l'utilisateur ne fournit que le nom du serveur Web auquel il souhaite se connecter (p. ex. www.dominio.es)[12].

Afin d'assurer une accessibilité et une disponibilité maximales de l'environnement Web, il est recommandé d'ouvrir également le port TCP/80[13], mais en configurant le serveur ou l'application Web pour qu'il effectue une redirection automatique (type 301 ou 302) de HTTP vers HTTPS pour toute requête (tentative de connexion) sur le port TCP/80. Plus précisément, il est recommandé d'utiliser une redirection 301 ("301 Moved Permanently") de HTTP vers HTTPS[14], car ce type de réponse peut être mis en cache et appliqué en permanence par les clients et navigateurs Web (voir section "3.7.3. Forcer l'utilisation (par défaut) de HTTPS").

[12] Ce n'est que lorsque, dans un avenir hypothétique, les navigateurs Web utiliseront par défaut le port TCP/443 (HTTPS), au lieu de HTTP, qu'il sera possible d'envisager la fermeture du port TCP/80 avec des garanties d'accessibilité élevées.

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

[14] https://support.google.com/webmasters/answer/6073543?hl=en

On peut aussi utiliser une mise en œuvre plus restrictive où le port TCP/80 reste fermé ou filtré, et où seul l'accès au port TCP/443 est disponible. Dans ce cas, pour l'accessibilité, toutes les références au serveur Web doivent spécifier le protocole HTTPS via "https://", par exemple, tous les liens dans "www.dominio.es" pointant vers "https://sede.dominio.es". Sinon, si le port TCP/80 est fermé et qu'un utilisateur tape simplement l'adresse (URL) du serveur ou de l'application Web dans la barre d'adresse de son navigateur Web, il obtiendra une erreur de connexion. Si vous n'avez pas le contrôle de toutes les références externes pointant vers le serveur Web (une situation courante lorsqu'il existe des références tierces pointant vers lui), ou en cas de doute, il est recommandé d'ouvrir le port TCP/80 et de configurer la redirection automatique de HTTP vers HTTPS décrite précédemment.

Ce scénario où il est recommandé d'avoir le port TCP/80 ouvert, avec une redirection automatique de HTTP à HTTPS, doit être utilisé même dans des scénarios plus sécurisés où HSTS est utilisé, et même si le domaine a été ajouté dans la liste préchargée au moyen de la directive "preload" (voir section "3.7.4. Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS"), car il peut se produire des situations où un client n'utilise pas la liste préchargée du HSTS, ou utilise une version du navigateur Web qui n'inclut pas encore le domaine étudié dans la liste préchargée.

3.1.1 Recommandations d'accès aux ports TCP/IP pour les services Web

Le résumé des recommandations d'accès aux ports TCP/IP associées aux services Web comprend :

- **Faites en sorte que le port standard TCP/443, associé au protocole HTTPS, soit ouvert.**
- **Ouvrez également le port TCP/80,** mais configurez le serveur ou l'application Web pour qu'il effectue une redirection automatique 301 ("301 Moved Permanently") de HTTP à HTTPS pour toute requête sur le port TCP/80.

3.2. Génération des clés cryptographiques

La génération adéquate des clés privées cryptographiques associées aux certificats numériques utilisés dans le *cadre* de HTTPS et, plus précisément, la sélection adéquate de l'algorithme de génération de clés et de la taille (ou longueur) de ces clés se relève fondamentale pour la sécurité de HTTPS. Il convient de noter que le point le plus faible lié aux clés est souvent associé à la gestion de celles-ci et à l'effort quotidien requis pour garder ces clés secrètes.

Les clés cryptographiques et leur lien avec le nom de l'entité, représentée par le serveur Web par le biais d'un certificat numérique, constituent l'identité cryptographique du serveur Web HTTPS.

L'algorithme le plus utilisé aujourd'hui pour la génération de clés privées, ainsi que des clés publiques qui leur sont associées, est le RSA (Rivest, Shamir et Adleman). Il est recommandé d'utiliser des clés RSA d'une longueur minimale de 2.048 bits (offrant 112 bits de sécurité). L'utilisation de clés plus grandes, par exemple celles de 3.072 bits, peut avoir un impact significatif sur les performances de l'environnement Web.

À l'avenir, il est fort probable que l'utilisation de clés ECDSA (*Elliptic Curve Digital Signature Algorithm*) —aussi appelées ECC (*Elliptic Curve Cryptography*)[15]— soit de plus en plus étendue par rapport au RSA, car elles offrent de meilleures performances que le RSA pour les clés possédant une longueur et un niveau de sécurité équivalents. Il est recommandé d'utiliser des clés ECDSA d'une longueur minimale de 256 bits (qui offrent une sécurité de 128 bits et une vitesse deux fois supérieure à celle d'une clé RSA de 2.048 bits, et environ six fois supérieure à celle d'une clé RSA équivalente de 3.072 bits).

Les clés cryptographiques et leur lien avec le nom de l'entité, représentée par le serveur Web par le biais d'un certificat numérique, constituent l'identité cryptographique du serveur Web HTTPS

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

Il convient toujours de rechercher un équilibre entre la sécurité et les performances et d'éviter de tomber dans l'excès en introduisant "trop" de sécurité. C'est pourquoi, à l'heure actuelle, pour les environnements Web standard, il est conseillé d'utiliser des clés RSA de 2.048 bits et des clés ECDSA de 256 bits. Du point de vue de la sécurité et des performances, ECDSA est préférable au RSA.

Il convient toujours de rechercher un équilibre entre la sécurité et les performances et d'éviter de tomber dans l'excès en introduisant "trop" de sécurité

À titre de référence, une clé RSA de 2.048 bits serait équivalente à une clé ECDSA de 224 bits (toutes deux avec une sécurité de 112 bits), une clé RSA de 3.072 bits à une clé ECDSA de 256 bits (toutes deux avec une sécurité de 128 bits), une clé RSA de 7.680 bits à une clé ECDSA de 384 bits (toutes deux avec une sécurité de 192 bits), et une clé RSA de 15.360 bits à une clé ECDSA de 512 bits (toutes deux avec une sécurité de 256 bits) . Le nombre de bits de sécurité serait à son tour équivalent à la taille recommandée pour les clés utilisées dans la cryptographie symétrique (p.ex. AES) et le nombre de bits ECDSA serait équivalent (approximativement) à la longueur des types de hash recommandés pour une robustesse similaire (p. ex. ECDSA 256 bits et SHA-256) .

Actuellement, le principal inconvénient des clés ECDSA est le manque de compatibilité avec certains navigateurs et clients Web (user agents), en particulier avec les plus anciens (p.ex. IE <= 8 ou Chrome sur Windows XP, ou OpenSSL < 1.0.0). Il est possible d'évaluer les algorithmes ou les suites cryptographiques qui sont acceptables pour un client Web spécifique, y compris les clés RSA et/ou ECDSA supportées dans le "SSL Client Test" de Qualys SSL Labs [Réf. - 3], ce qui permet de sélectionner chacun des clients et navigateurs Web pour vérifier toutes leurs fonctionnalités liées à HTTPS [Réf. - 6], ou celles qui sont supportées dans "SSL Cipher Suite Details of Your Browser" [Réf. - 5].

Il est possible d'utiliser simultanément des clés RSA et ECDSA, afin de fournir le plus haut niveau de sécurité possible pour chaque type de client Web. Cependant, cette possibilité dépend des technologies et des plateformes utilisées dans la mise en œuvre de HTTPS. Il est donc recommandé de vérifier l'existence d'un support pour l'utilisation simultanée de différentes clés dans l'environnement Web où le présent rapport sera appliqué. Par exemple, Apache permet ce type de configuration, en utilisant simultanément différentes clés RSA et ECDSA et, par conséquent, différents certificats numériques.

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

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

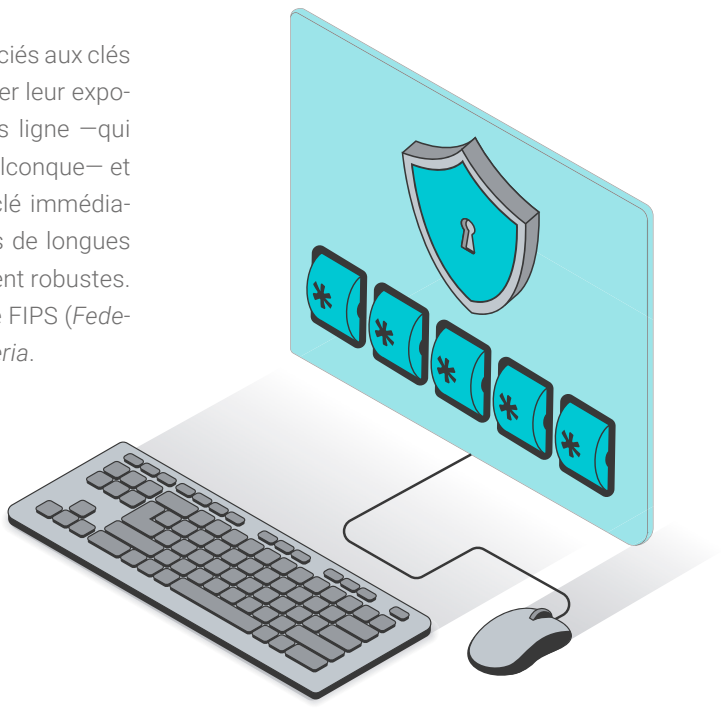
Lors de la sélection et de la génération des clés, il est nécessaire d'évaluer les options qui sont raisonnablement sûres à l'heure actuelle, et qui continueront à l'être jusqu'au remplacement de la clé, car elles permettront de déterminer la période pendant laquelle les clés resteront raisonnablement protégées avant d'être remplacées.

Lors de la sélection et de la génération des clés, il est nécessaire d'évaluer les options qui sont raisonnablement sûres à l'heure actuelle, et qui continueront à l'être jusqu'au remplacement de la clé

Il convient de garder à l'esprit que dans la plupart des attaques et des incidents de sécurité liés à la cryptographie, les attaquants préfèrent contourner le système de chiffrement —plutôt que de le casser ou de le compromettre— car il s'agit d'une tâche beaucoup plus simple. Il est donc recommandé de gérer correctement les clés, en conservant les clés privées secrètes à tout moment, en utilisant un bon générateur de nombres pseudo-aléatoires (PRNG), en protégeant les clés avec un mot de passe fort (phrase de passe) et en les stockant de manière sécurisée (p.ex. en utilisant un module de sécurité matériel HSM), en faisant des sauvegardes sécurisées des clés et en renouvelant les clés périodiquement.

En aucun cas, le processus de génération de la clé privée ne doit être délégué à l'Autorité de Certification (désormais AC). Au lieu de cela, elle doit être générée localement et fournir à l'AC une demande de signature de certificat (CSR, Certificate Signing Request), afin que l'AC puisse émettre le certificat numérique qui lui est associé. La CSR doit être générée en utilisant au moins SHA-256 comme algorithme de signature ou *hachage*. En outre, le principe du moindre privilège doit être appliqué, en exposant la clé privée au plus petit nombre possible de systèmes et de personnes.

Afin de garantir une génération aléatoire des nombres associés aux clés cryptographiques aussi bonne que possible, et de minimiser leur exposition, il est recommandé d'utiliser des équipements hors ligne —qui ne soient pas exposés à un réseau de communication quelconque— et dotés d'une entropie suffisante (p.ex. ne pas générer la clé immédiatement après le redémarrage de l'équipement, mais après de longues périodes d'utilisation), afin de générer des clés suffisamment robustes. Idéalement, l'équipement devrait être conforme à la norme FIPS (*Federal Information Processing Standard*) ou aux *Common Criteria*.



Le stockage des clés doit être protégé par un mot de passe fort (phrase de passe) et, de préférence, par des modules de sécurité matériels spécifiques (HSM, *Hardware Security Module*), au lieu d'avoir une ou plusieurs copies des clés dans des systèmes de fichiers standard sur un ou plusieurs serveurs. De même, l'utilisation d'un même ensemble de clés et de leurs certificats numériques associés sur des serveurs Web présentant des niveaux de sécurité et de criticité différents est très problématique, car un incident de sécurité sur l'un des serveurs exposerait également le reste des serveurs.

Enfin, les clés doivent être correctement gérées, en enregistrant le moment où elles sont générées, mises en production et retirées ou renouvelées, à l'aide d'un effacement sécurisé. Les clés doivent être renouvelées périodiquement (et si possible automatiquement) pour éviter qu'elles soient utilisées pendant de très longues périodes et pour minimiser l'impact en cas d'incident de sécurité ou de tout autre événement susceptible de les exposer. Après un incident de sécurité, il convient de révoquer les certificats numériques précédents, de générer des nouvelles clés et de demander des nouveaux certificats numériques.

Une pratique courante consiste à renouveler les clés cryptographiques à chaque fois que le certificat numérique associé est renouvelé (voir section "3.3.5. Renouvellement des certificats numériques"), bien que l'introduction de nouveaux mécanismes de sécurité tels que HPKP (voir section "3.3.9 Restriction des certificats numériques légitimes : HPKP") peut conditionner la période et/ou la procédure de renouvellement des clés cryptographiques (car les épingles codes sont associés à la clé cryptographique et doivent être mis à jour si la clé est renouvelée).

Le stockage des clés doit être protégé par un mot de passe fort (phrase de passe) et, de préférence, par des modules de sécurité matériels spécifiques (HSM, *Hardware Security Module*)

3.2.1 Recommandations pour la génération de clés cryptographiques

Le résumé des recommandations pour la génération de clés cryptographiques comprend :

- **Utilisez des clés RSA de 2.048 bits.**
- Alternativement, ou de manière complémentaire, **utilisez des clés ECDSA de 256 bits,** car elles offrent une meilleure sécurité et de meilleures performances que les clés RSA.
- Il convient toujours de **rechercher un équilibre entre la sécurité et les performances et d'éviter de tomber dans l'excès en introduisant "trop" de sécurité**. Il est donc préférable, à l'heure actuelle, d'utiliser des clés ECDSA (256 bits) plutôt que RSA (2.048 bits).
- **Utiliser les clés ECDSA et RSA simultanément si les technologies Web employées le permettent.**
- **Utilisez un bon générateur de nombres aléatoires pour générer les clés.**
- **Le stockage des clés doit être protégé par un mot de passe fort (phrase de passe)** et, alternativement, par des modules de sécurité matériels spécifiques (HSM, *Hardware Security Module*).
- **Gérez les clés de façon correcte :** conserver le secret des clés privées, disposez de sauvegardes sécurisées des clés, renouvelez les clés périodiquement, faire un effacement sécurisé, renouvelez les clés à la suite d'un incident de sécurité, exposez les clés privées au plus petit nombre possible de systèmes et de personnes, etc.

3.3. Gestion des certificats numériques

Cette section comprend les recommandations relatives à la sélection, l'obtention et la gestion des certificats numériques utilisés par HTTPS et liés aux clés cryptographiques mentionnées précédemment (voir section "3.2. Génération des clés cryptographiques"). Le fait de disposer de clés cryptographiques et de certificats numériques robustes et sécurisés empêche un attaquant potentiel de supplanter facilement l'environnement Web.

Le fait de disposer de clés cryptographiques et de certificats numériques robustes et sécurisés empêche un attaquant potentiel de supplanter facilement l'environnement Web

3.3.1 Types de certificats numériques

Il existe actuellement trois types de certificats numériques, classés en fonction du niveau de sécurité ou de confiance qu'ils offrent et de leur coût : certificats à validation de domaine (DV, Domain Validated), certificats à validation de l'organisation (OV, Organisation Validated) et certificats à validation étendue (EV, Extended Validation).

Les certificats DV sont validés automatiquement et sont les moins chers, tandis que les certificats EV se soumettent à des processus de validation standardisés (par le CAB Forum ou le CA/Browser Forum[18]), ils sont plus chers et ils sont traités différemment par les navigateurs Web (les informations concernant l'organisation propriétaire du certificat numérique —raison sociale ou pays d'origine— apparaissent dans la barre d'adresse ou URL affichée en vert).

L'utilisation d'un type de certificat numérique ou d'un autre dépend de multiples facteurs, dont des facteurs économiques ainsi que ceux liés à l'image de l'organisation et à la confiance qu'elle souhaite transmettre aux utilisateurs du serveur ou de l'application Web.

[18] https://cabforum.org

3.3.2 Délivrance des certificats numériques : Les Autorités de Certification (AC)

Le présent rapport est centré sur la mise en œuvre du protocole HTTPS sur les serveurs Web publics. Il est donc obligatoire de disposer d'un certificat numérique émis par une Autorité de Certification publique fiable et largement reconnue. Il n'est donc pas possible, pour une mise en œuvre correcte de HTTPS, d'utiliser des certificats numériques auto-signés ou signés par une AC privée.

Suivant le Schéma National de Sécurité (ENS), il est recommandé d'utiliser un prestataire de services de confiance.

Suivant le Schéma National de Sécurité (ENS), il est recommandé d'utiliser un prestataire de services de confiance

Même dans le cas des serveurs Web internes, il est recommandé d'avoir un certificat numérique émis par une AC publique de confiance ou, à défaut, par une AC privée qui soit également largement reconnue par les clients Web.

Lorsque vous demandez un certificat numérique à une AC publique, il faut être conscient que vous établissez avec elle une relation étroite à moyen ou long terme, surtout si vous utilisez de nouveaux mécanismes de sécurité tels que HPKP (voir section "3.3.9 Restriction des certificats numériques légitimes : HPKP").

Il est donc recommandé d'évaluer et de sélectionner l'Autorité de Certification sur la base de critères objectifs, en fonction de la nature, de la criticité et du niveau de sécurité du serveur Web où le HTTPS doit être utilisé. Ces critères devraient inclure le niveau de consolidation et de maturité de l'AC, sa réputation dans l'industrie, la reconnaissance par plusieurs navigateurs et clients Web de leur confiance envers l'AC, ses capacités à gérer de nombreux certificats numériques, le type de service d'assistance administrative et technique fourni par l'AC, sa position dans l'industrie et sa volonté d'adopter de nouvelles technologies et normes (telles que : CT —voir section "3.3.6. *Certificate Transparency* (CT)"—, CAA —voir section "3.3.7. CAA (*Certification Authority Authorization*) "—, *OCSP Stapling* (Agrafage OCSP) —voir section "3.6.3. *OCSP Stapling*"—, la possibilité d'obtenir des certificats numériques basés sur des clés RSA et ECDSA —voir section "3.2. Génération des clés cryptographiques"—, etc.) et, bien sûr, le niveau de sécurité et les mécanismes de protection de l'AC elle-même et sa transparence dans la gestion des incidents de sécurité (sur la base de l'historique des incidents passés).

Les mécanismes de révocation des certificats numériques disponibles constituent un autre élément à prendre en compte lors de l'évaluation de l'AC que l'on prétend choisir (voir section "3.6. Mécanismes de révocation des certificats numériques"). Ces mécanismes doivent pouvoir vérifier le statut de révocation par CRL ou par OCSP et offrir la possibilité d'analyser la disponibilité et les performances.

Comme alternative aux AC commerciales, il existe actuellement une AC à but non lucratif appelée Let's Encrypt [Réf. - 11], qui vise à généraliser l'utilisation de HTTPS sur Internet en délivrant des certificats numériques gratuits (de type DV). Let's Encrypt représente une alternative valable pour obtenir des certificats numériques légitimes et largement reconnus. Une de ses caractéristiques principales est qu'il s'agit d'un modèle qui réduit la période de renouvellement des certificats (actuellement fixée à 90 jours, c.-à-d. 3 mois[19]) afin de minimiser l'impact d'un éventuel incident de sécurité et de promouvoir l'automatisation du processus de demande et de renouvellement des certificats numériques.

Pour information[20], en février 2016, Let's Encrypt a introduit de nouveaux supports pour la génération de certificats numériques basés sur des clés ECDSA[21], et en septembre 2017 a annoncé une nouvelle AC racine et plusieurs AC intermédiaires ECDSA, fournissant ainsi une chaîne de certificats (ou de confiance) complète basée sur ECDSA.

Une fois qu'elle a émis les certificats numériques, l'AC inclut également sa propre signature sur le certificat pour prouver sa validité. Pour ce faire, elle emploie un algorithme de signature comme SHA-256 ou d'autres plus robustes (p. ex. SHA-384, SHA-512...), également référencés comme SHA-2. En aucun cas, les algorithmes SHA-1 ou MD5 ne doivent être utilisés pour signer le certificat.

Étant donné que la demande d'un certificat via une CSR ne permet pas de spécifier l'algorithme de signature à utiliser par l'AC, avant de demander le certificat numérique, il est recommandé de vérifier au-

Il existe actuellement une AC à but non lucratif appelée Let's Encrypt [Réf. - 11], qui vise à généraliser l'utilisation de HTTPS sur Internet en délivrant des certificats numériques gratuits

[19] https://letsencrypt.org/2015/11/09/why-90-days.html

[20] https://scotthelme.co.uk/ecdsa-certificates/

[21] https://letsencrypt.org/upcoming-features/

près de l'AC quels sont les algorithmes qu'elle utilise actuellement pour signer les certificats numériques. En outre, il est également conseillé de vérifier auprès de l'AC quelle sera la chaîne de certification (ou de certificats) complète associée au certificat numérique demandé, c'est-à-dire la liste complète des AC intermédiaires liées à la délivrance du certificat jusqu'à l'AC racine, et d'identifier l'AC racine elle-même.

Il est important de préciser que la recommandation de ne pas utiliser SHA-1 s'applique à la signature des certificats numériques racine des AC. Les algorithmes de signature utilisés dans le passé, tels que MD5 ou SHA-1, sont considérés comme peu sûrs et ne doivent pas être utilisés pour signer des certificats numériques :

- **MD5** fa été conçu pour fournir une sécurité de 64 bits et a été complètement violé, de manière pratique sur les certificats numériques utilisés pour HTTPS, en 2009[22].

- **SHA-1** a été conçu pour fournir 80 bits de sécurité, mais il fournit réellement 61 bits de sécurité effective. Depuis janvier 2016, les AC ne doivent plus émettre de certificats numériques signés avec SHA-1[23] (à la demande initiale de Microsoft) et en janvier/février 2017, les principaux navigateurs Web (Firefox 51[24], Edge & IE 11[25], ou Chrome 56[26]) ont commencé à rejeter les certificats numériques signés avec SHA-1, jugés peu fiables.

  En février 2017, Google, en collaboration avec l'Institut CWI d'Amsterdam, a annoncé être parvenu à générer une collision via la fonction SHA-1 [Réf. - 12].

Il est important de préciser que la recommandation de ne pas utiliser SHA-1 s'applique à la signature des certificats numériques racine des AC

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-publicweb/

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

3.3.3 Entité associée au certificat numérique

Si nous analysons l'intérieur des champs ou des éléments de sécurité inclus dans un certificat numérique (voir section "3.3.4. Validité du certificat numérique"), le nom ou l'identifiant de l'entité signataire (environnement, serveur ou application Web) relève d'une importance particulière.

Lorsqu'un navigateur ou un client Web se connecte via HTTPS au serveur Web, il vérifie que le nom de l'entité associée au certificat numérique correspond au nom utilisé dans l'adresse Web (ou URL) visitée. Dans le cas contraire, des erreurs de certificat seront générées, ce qui risque de confondre les utilisateurs et de réduire leur confiance envers l'environnement Web.

Il est donc essentiel de s'assurer que les noms et entités contenus dans le certificat numérique correspondent et assurent une couverture (représentent) sur tous les serveurs Web utilisant ce certificat, soit par une référence directe ou unique (telle que www.dominio.es), soit par une liste complète des noms des serveurs Web (p. ex. www.dominio.es, portal.dominio.es et sede.dominio.es), soit par un certificat numérique *Wildcard* (Joker) couvrant tous les sous-domaines du domaine principal (tel que *.dominio.es)[27]. Lors de l'utilisation d'un certificat *Wildcard*, il faut surtout prendre en compte que celui-ci ne doit être utilisé que sur des serveurs Web présentant le même niveau de sécurité, afin de minimiser l'impact d'un éventuel incident de sécurité (et l'accès aux clés privées liées au certificat numérique). Là encore, le principe du moindre privilège doit être appliqué, en essayant de limiter le partage du certificat numérique et de la clé privée entre de nombreux départements, systèmes et individus.

Bien qu'il soit également possible d'obtenir des certificats numériques multi-domaines, c'est-à-dire valables pour des serveurs Web de différents domaines, pour la même raison (favoriser leur utilisation dans des environnements présentant le même niveau de sécurité et limiter le partage du certificat numérique et de la clé privée), il est recommandé de séparer les certificats numériques pour chaque domaine.

Si nous analysons l'intérieur des champs ou des éléments de sécurité inclus dans un certificat numérique, le nom ou l'identifiant de l'entité signataire relève d'une importance particulière

[27] L'utilisation de certificats numériques Wildcard est assez fréquente dans les CDN qui gèrent le trafic HTTPS entre plusieurs serveurs Web au sein d'une même organisation.

En revanche, il est recommandé de s'assurer que le certificat numérique comprend à la fois le nom du serveur Web représenté (ou la liste des serveurs Web ou le certificat *Wildcard*) et celui de son domaine, c'est-à-dire www.dominio.es et dominio.es (sans le préfixe "www") ou *. dominio.es et dominio.es (dans le cas de l'utilisation d'un certificat *Wildcard*). Ce scénario est assez fréquent pour les certificats numériques actuels (mais doit être vérifié explicitement), car les deux noms font généralement référence —dans le service DNS— à la même adresse IP, et donc au même serveur. En règle générale, le certificat numérique doit être valable pour tous les noms existant dans le service DNS pour le serveur Web associé.

Il est recommandé de s'assurer que le certificat numérique comprend à la fois le nom du serveur Web représenté et celui de son domaine

Dans le passé, le nom de l'entité associée au certificat numérique était indiqué dans le champ "CN" (*Common Name* ou *commonName*) du certificat. Par la suite, la norme RFC 2818[28] a suggéré d'identifier le nom par le champ "SAN" (*Subject Alternative Name*, extension "subjectAltName" de type "dNSName") et, en son absence, par le "CN". Cette RFC, publiée en mai 2000, donnait déjà la priorité au "SAN" sur le "CN", venant même à sous-estimer ce dernier. Malgré cela, de nombreux clients Web ont maintenu les deux champs au cours des dernières années.

Cependant, à partir de Firefox 48[29] et Chrome 58[30] il n'y a plus de support pour identifier le nom via le "CN", ce qui fait qu'une erreur de certificat se présente dans le cas où le nom du serveur Web ne soit pas disponible dans le champ "SAN". Il est donc recommandé d'utiliser uniquement le champ "SAN" pour inclure la liste des noms des entités associées au certificat numérique (ou représentées par celui-ci).

.

[28] https://tools.ietf.org/html/rfc2818

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

[30] https://www.chromestatus.com/features/4981025180483584

3.3.4 Validité du certificat numérique

Les certificats numériques X.509 contiennent de nombreux champs ou éléments de sécurité. Ceux-ci sont évalués par un navigateur ou un client Web pour déterminer la validité du certificat. Pour cette raison, il est nécessaire de s'assurer à tout moment que toutes leurs valeurs sont valides, notamment :

- **Le nom de l'entité ou du serveur Web** (voir section "3.3.3. Entité associée au certificat numérique").
- **La période de validité ou d'expiration du certificat** (avec la date de début et de fin de cette période).
- **L'AC qui a émis et signé le certificat** (voir section "3.3.2. Délivrance des certificats numériques : Les Autorités de certification (AC)").
- **Révocation du certificat** (voir section "3.6. Mécanismes de révocation des certificats numériques").
- **Objectifs du certificat :** authentification du serveur (serveurs Web qui font l'objet du présent rapport), authentification du client, signature de code des fichiers exécutables ou du logiciel, courrier électronique sécurisé, etc. (cette classification est clairement visible, surtout dans les environnements Windows).

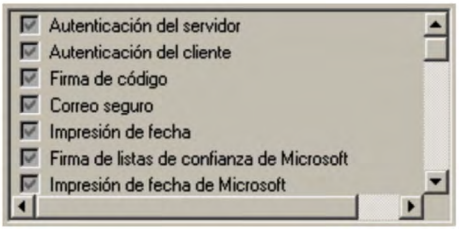


Figure 3.
Les différents objectifs de l'utilisation d'un certificat numérique:
Authentification du serveur,
Authentification du client,
Signature de code,
Courrier sécurisé,
Horodatage,
Signature des listes dignes de confiance de Microsoft,
Horodatage de Microsoft.
Source : Microsoft

Plus précisément, lors de l'évaluation de la validité d'un certificat numérique X.509, il est également nécessaire de vérifier que l'ensemble de la chaîne de certification est valide et, plus précisément, qu'elle n'est pas incomplète. Il est recommandé que le serveur Web fournisse aux clients Web la chaîne de certification complète (leur évitant ainsi d'avoir à la résoudre et à la reconstruire de manière autonome, une situation qui peut conduire à des erreurs), en fournissant chacun des certificats numériques émis ou associés à l'AC dans l'ordre correct, c'est-à-dire de l'AC intermédiaire qui suit immédiatement l'AC racine jusqu'au certificat numérique du serveur Web. La chaîne de certification ne doit inclure que les certificats numériques nécessaires ; pour des raisons de rendement, il n'est pas recommandé d'ajouter le certificat de l'AC racine à la chaîne, car il devrait déjà être reconnu par les navigateurs et clients Web (vu qu'il est disponible dans leur magasin de certificats racine).

Il existe certains outils Web, tels que "Certificate Analyser"[31], qui permettent d'évaluer la chaîne de certification du serveur Web.

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

3.3.5 Renouvellement des certificats numériques

La durée de validité habituelle ou de renouvellement des certificats est actuellement de 1, 2 ou 3 ans, selon le type de certificat (DV, OV ou EV). Logiquement, la période à choisir est influencée par le coût économique (les AC offrant généralement des réductions pour les périodes plus longues) et la fréquence à laquelle les procédures administratives de renouvellement associées doivent être effectuées.

Il est recommandé de renouveler les certificats numériques chaque année, et même plus fréquemment s'il est possible d'automatiser le processus de renouvellement (voir section "3.3.2. Délivrance des certificats numériques : Les Autorités de certification (AC)" et les références à Let's Encrypt). Si la révocation complète, efficace et fiable d'un certificat numérique compromis s'avère très complexe, d'un point de vue pratique, les certificats numériques ayant une durée de vie plus courte seraient considérés comme plus sûrs.

En mars 2017, un processus de vote a été lancé au sein du Forum CAB (Ballot 193) pour définir une nouvelle norme stipulant que tous les certificats numériques (émis à partir du 1er mars 2018) auraient une durée de validité maximale de 825 jours[32] (environ 27 mois, soit 2 ans et trois mois), face à la durée maximale de 39 mois (soit 3 ans et trois mois) de l'époque. Le résultat du vote a ratifié la nouvelle durée de validité maximale des certificats numériques (émis à partir du 1er mars 2018) et a permis de la fixer à 825 jours[33].

Les certificats numériques doivent être renouvelés périodiquement et toujours avant leur expiration. Chaque fois que le certificat numérique est renouvelé, il est également recommandé de renouveler la chaîne de certification complète, y compris toutes les AC intermédiaires (dont les certificats numériques peuvent également avoir expiré), afin de fournir aux clients Web une liste complète et actualisée de la chaîne de certification. Il est également conseillé de renouveler les clés cryptographiques à chaque renouvellement du certificat numérique associé, à moins que HPKP soit utilisé (voir section "3.3.9 Restriction des certificats numériques légitimes : HPKP").

La durée de validité habituelle ou de renouvellement des certificats est actuellement de 1, 2 ou 3 ans, selon le type de certificat (DV, OV ou EV)

[32] https://cabforum.org/pipermail/public/2017-March/010024.html

[33] https://cabforum.org/pipermail/public/2017-March/010098.html

Une fois renouvelé, il est recommandé de déployer le nouveau certificat numérique environ une semaine après l'avoir obtenu, en essayant d'éviter des problèmes avec les clients Web dont la référence temporelle n'a pas été configurée correctement et de donner à l'AC suffisamment de temps pour propager le nouveau certificat dans son service OCSP. En outre, il convient de tenir compte du fait qu'il n'est pas conseillé de révoquer le certificat numérique précédent immédiatement après son renouvellement, mais plutôt de prévoir le chevauchement des périodes de validité des deux certificats pour permettre un processus de migration en douceur de l'un à l'autre.

3.3.6 *Certificate Transparency* (CT)

La norme *Certificate Transparency* (CT) de Google [Réf. - 13], ou Transparence des Certificats, établit un environnement ouvert (ou *framework*) qui permet aux AC de contrôler et vérifier le processus d'émission des certificats numériques en quasi-temps réel. La proposition initiale est en train de devenir une norme industrielle, associée à la norme RFC 6962 [Réf. - 14].

La norme *Certificate Transparency* (CT) de Google, ou Transparence des Certificats, établit un environnement ouvert (ou *framework*) qui permet aux AC de contrôler et vérifier le processus d'émission des certificats numériques en quasi-temps réel

Ces dernières années, les AC se sont révélées vulnérables à l'utilisation inappropriée et à la manipulation de leurs procédures de délivrance de certificats numériques. Le cadre CT permet d'identifier l'émission de nouveaux certificats, puisque ceux-ci doivent être répertoriés par les AC dans un registre publique vérifiable, et dont l'intégrité ne peut être manipulée, puisqu'il fait appel à des mécanismes cryptographiques qui permettent uniquement d'ajouter des registres (sans les effacer ou modifier)[34]. Le CT permet d'identifier rapidement les scénarios indésirables : lorsqu'une AC émet un certificat par erreur, lorsqu'une AC (qui a été rachetée par une autre AC) émet des certificats non sollicités ou lorsqu'une AC compromise ou qui devient malveillante émet des certificats pour supplanter certains sites Web. La détection précoce atténue l'impact des certificats illégitimes et fournit, à son tour, de la transparence et des capacités publiques pour contrôler la santé et l'intégrité de l'ensemble de l'écosystème mondial de l'infrastructure à clé publique (PKI) dans l'univers d'Internet.

[34] Les "arbres de hachage de Merkle" sont utilisés à cet effet : www.certificate-transparency.org/log-proofs-work.

Une fois que l'émission d'un certificat numérique non légitime a été identifiée par le CT, il est possible de mener des actions pour révoquer le certificat dès que possible (voir section "3.6. Mécanismes de révocation des certificats numériques") et, si nécessaire, le remplacer par un nouveau certificat numérique légitime.

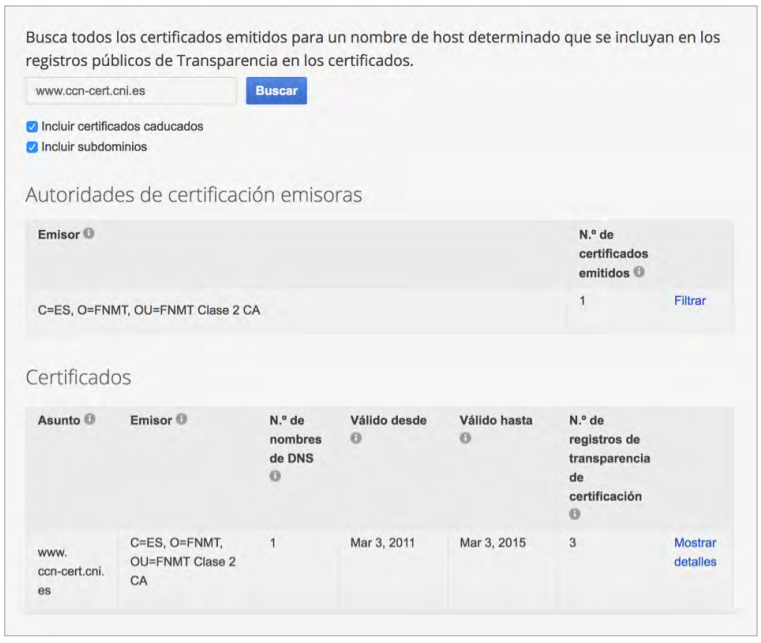
Le CT permet de surveiller les journaux (*logs*) publics et d'identifier l'émission de certificats numériques pour ses propres domaines sans l'autorisation du propriétaire du domaine, soit par des AC différentes à celle qui est utilisée, soit par l'AC elle-même. Pour ce faire, il existe des ressources [Réf. - 15][35] qui permettent de consulter les registres existants pour un domaine, voire pour un nom (ou un hôte) spécifique, y compris les certificats numériques qui ont déjà expiré et ceux associés à des sous-domaines. La réponse obtenue permet d'identifier les AC émettrices et les certificats enregistrés dans les journaux CT publics.

À titre de référence, le 13 décembre 2016, près de 172 millions d'entrées ont été enregistrées dans les *logs* (journaux) de CT, tandis que le 16 mars 2017, ce nombre est passé à près de 249 millions d'entrées.

Parallèlement, il existe des services de surveillance qui permettent de savoir si un certificat est émis pour un domaine sans la connaissance de l'entité légitime, comme Computest Suricat (https://suricat.io) ou Cert Spotter (https://sslmate.com/certspotter/).

Une fois que l'émission d'un certificat numérique non légitime a été identifiée par le CT, il est possible de mener des actions pour révoquer le certificat dès que possible et, si nécessaire, le remplacer par un nouveau certificat numérique légitime

Figure 4.
Résultats de la requête CT pour www.ccn-cert.cni.es. Source : Google



[35] https://www.entrust.com/ct-search/

Le navigateur Web Chrome exige que tous les certificats de type EV soient disponibles dans le CT à partir du 1er janvier 2015[36] et, en raison des irrégularités commises par Symantec en tant qu'AC, la version Chrome 53 exige (à partir d'octobre 2015) que tous les certificats numériques émis par Symantec après le 1er juin 2016[37] soient utilisés par CT. En outre, à partir d'octobre 2017, Chrome exige le CT pour tous les nouveaux certificats numériques[38].

L'environnement CT est composé de trois éléments principaux[39] :

- **Journaux ou registres de certificats :**
  Service chargé de maintenir ensemble les registres cryptographiques publics et les certificats numériques, où seuls des registres peuvent être ajoutés (généralement fournis par les AC).
- **Moniteurs :**
  Service qui se connecte aux registres publics à la recherche de certificats suspects.
- **Auditeurs :**
  Service qui, d'une part, vérifie que les journaux publics fonctionnent correctement et sont cohérents (sur le plan cryptographique) et, d'autre part, qui peut vérifier l'existence d'un certificat spécifique dans le registre CT. Les navigateurs Web finiront par mettre en œuvre ces fonctionnalités.

Pendant l'établissement d'une session TLS, les serveurs Web HTTPS peuvent fournir des informations CT aux clients Web à travers leurs réponses, en utilisant trois méthodes : l'extension de certificat X.509v3 (fournie par l'AC), l'extension TLS (disponible, p. ex., dans Apache via le module "mod_ssl_ct"[40]) ou l'*OCSP Stapling*[41] (si le protocole est supporté par l'AC).

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

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

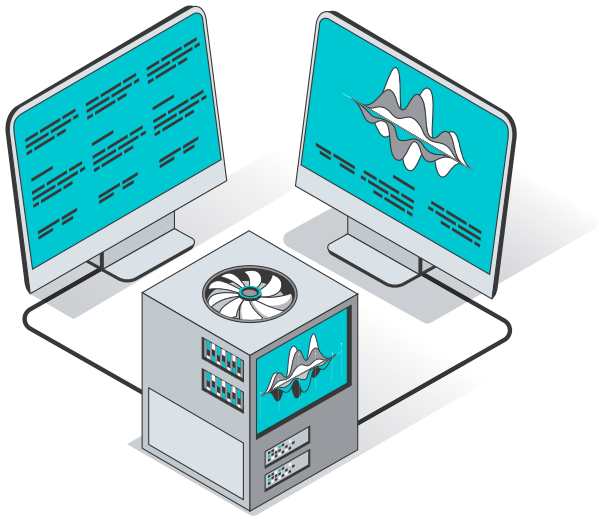
41 http://www.certificate-transparency.org/resources-for-site-owners

La méthode la plus simple de communication CT à partir du serveur Web consiste à utiliser une AC émettrice de certificats X.509 datés et qualifiés au moyen de l'horodatage de la signature (*timestamp*) et contenant la preuve qu'ils ont été déposés dans plusieurs journaux CT valides (*SCT, Signed Certificate Timestamp*)[42]. Toutefois, il est préférable d'utiliser une nouvelle extension TLS dédiée à cet effet ou l'*OCSP Stapling* (à condition que l'AC prenne en charge les SCT), car ces deux méthodes permettent d'inclure de nouvelles valeurs SCT pour de nouveaux journaux (en s'ajoutant ou en remplaçant les valeurs précédentes) sans devoir renouveler le certificat numérique.

La méthode la plus simple de communication CT à partir du serveur Web consiste à utiliser une AC émettrice de certificats X.509

En outre, en janvier 2017, Chrome a proposé[43] la création de l'en-tête HTTP "*Expect-CT*" [Réf. - 16], ainsi nommé pour indiquer que CT est pris en charge par le serveur Web et que le client doit s'attendre à recevoir une valeur SCT valide. Comme pour les autres en-têtes HTTP de sécurité tels que HPKP (voir section "3.3.9 Restriction des certificats numériques légitimes : HPKP") ou CSP (voir section "3.7.6. *Content Security Policy* (CSP)"), "*Expect-CT*" peut être utilisé de manière informative en faisant que le navigateur Web envoie un rapport à l'URL spécifiée lorsque la transparence du certificat valide n'est pas reçue (directive "report-uri"), pour détecter l'absence de SCT valides ou, de manière stricte, en bloquant les connexions HTTPS associées à un certificat non conforme à la politique CT. Pour cela, "*Expect-CT*" n'utilise pas deux en-têtes HTTP différents, comme dans le cas de HPKP ou CSP, mais utilise la directive facultative "enforce" dans l'en-tête de réponse.

Vu que Chrome exigera le CT pour tous les nouveaux certificats en octobre 2017, cet en-tête permet de protéger également les anciens certificats (émis avant octobre 2017) ou les nouveaux certificats émis par une AC malveillante avec des dates antérieures.



[42] https://www.certificate-transparency.org/how-ct-works

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

3.3.7 CAA (*Certification Authority Authorization*)

La norme CAA (*Certification Authority Authorization*) [Réf. - 17] établit un nouveau type de registre dans le service de résolution de noms DNS (RR, *Resource Record*, de type 257) qui permet au propriétaire d'un domaine de spécifier le nom de l'AC ou des AC autorisée(s) à émettre des certificats numériques pour ce domaine (ce qui signifie que les autres AC ne sont pas autorisées à le faire).

Par conséquent, la CAA associe le service DNS au processus de délivrance des certificats numériques utilisés, par exemple, dans le protocole HTTPS, ce qui permet d'utiliser le service DNS comme mécanisme de validation externe associé au protocole HTTPS et, en particulier, à la gestion et à la délivrance des certificats numériques.

La CAA permet à une AC (bien intentionnée) d'ajouter des contrôles (administratifs) supplémentaires afin de réduire le risque de délivrer des certificats numériques de manière involontaire et contraire aux stipulations du propriétaire du domaine Web.

La CAA permet à une AC d'ajouter des contrôles supplémentaires afin de réduire le risque de délivrer des certificats numériques de manière involontaire et contraire aux stipulations du propriétaire du domaine Web

En février 2017, un processus de vote a été lancé dans le forum de l'ACR (Ballot 187) pour rendre la norme CAA obligatoire pour les AC[44] (selon cette norme, les AC doivent vérifier et contrôler les enregistrements CAA du DNS avant de délivrer un nouveau certificat numérique). Le résultat du vote a ratifié que la vérification des enregistrements CAA est obligatoire pour les AC[45] à partir du 8 septembre 2017 [Réf. - 18]. Si la configuration des CAA ne permet pas à l'AC d'émettre des certificats numériques pour ce domaine, le processus d'émission doit être interrompu (et potentiellement revu manuellement par le personnel de l'AC).

Ces contrôles apportent des améliorations significatives à l'infrastructure à clé publique (PKI) de l'Internet, qui se compose de toutes les AC publiques de confiance, en empêchant que l'écosystème et la confiance envers les AC publiques deviennent aussi faibles que l'AC la plus faible.

Les enregistrements CAA fournissent une granularité permettant de déterminer les AC associées au domaine ("issue"), les AC associées aux certificats *Wildcard* ("issuewild") —voir section "3.3.3. Entité associée au certificat numérique"— et fournir des capacités de notification en cas d'erreurs ou de demandes de certificats invalides ("iodef") via une URL ou via une adresse électronique.

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

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

Le format des enregistrements CAA comprend le domaine associé aux certificats numériques et qui établit la restriction dans le DNS, ainsi que le nom de domaine de l'AC autorisée à émettre des certificats numériques pour ce domaine[46] :

```
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"
```

La norme CAA est suffisamment souple et permet de définir des propriétés supplémentaires futures au moyen de paires de noms (de variable) et de valeurs.

3.3.8 DNSSEC et DANE

DNSSEC (*Domain Name System Security Extensions*) est un ensemble de technologies qui ajoutent de l'intégrité au système de noms de domaine, DNS (*Domain Name System*). Aujourd'hui, un attaquant actif sur un réseau peut intercepter n'importe quelle requête DNS et modifier arbitrairement sa réponse. Avec le DNSSEC, toutes les réponses peuvent être tracées de manière cryptographique jusqu'à la racine du DNS.

D'autre part, l'authentification des entités nommées basée sur le DNS (DANE) est une norme distincte qui s'appuie sur les bases créées par le DNSSEC et établit des liens entre le DNS et le TLS. DANE pourrait être utilisé pour renforcer la sécurité de l'écosystème PKI existant basé sur les AC ou pour fournir une alternative différente.

Bien qu'il n'y ait pas de consensus sur la question de savoir si le DNSSEC est le chemin correct vers lequel devrait se diriger l'Internet, sa mise en place prend forme et continue de s'améliorer. Les navigateurs ne prennent pas encore en charge DNSSEC et DANE, mais il semble qu'ils commencent à être utilisés pour améliorer la sécurité du système de courrier électronique.

[46] La valeur 128 ajoute un drapeau de criticité pour cette politique DNS, ce qui permettra d'ajouter de nouvelles sémantiques dans les futures versions des CAA, et d'indiquer que la politique doit être comprise par l'AC afin de pouvoir traiter les enregistrements CAA associés et procéder à l'émission du certificat numérique.

3.3.9 Restriction des certificats numériques légitimes : HPKP

L'en-tête de sécurité du protocole HTTP connu sous le nom de HPKP ou PKP (*HTTP Public Key Pinning*) [Réf. - 19] permet au serveur Web de déclarer qu'un certificat numérique pour ce serveur ne doit être considéré comme valide que s'il est inclus dans la liste spécifique de certificats (épingles codes) énumérés dans cet en-tête. Par conséquent, les navigateurs et clients Web prenant en charge HPKP ne se connecteront pas à l'environnement Web si le certificat numérique reçu (ou tout autre certificat de la chaîne de certification, p. ex. ceux des AC intermédiaires) ne correspond pas aux épingles reçues.

Cet en-tête de sécurité est tombé en désuétude et n'est pas recommandé actuellement (en 2022) en raison, entre autres, de risques tels que *HPKP Suicide* ou *RansomPKP*. Certains navigateurs peuvent encore le prendre en charge pour des raisons de compatibilité. Cet en-tête a été remplacé par le CT (*Certificate Transparency*).

HPKP est probablement l'un des mécanismes de sécurité les plus stricts disponibles de nos jours, et il introduit le risque de bloquer l'accès à l'environnement Web s'il n'est pas configuré correctement. Il doit donc être mis en œuvre avec prudence et soumis à une évaluation minutieuse. En particulier, il doit être utilisé si l'on considère qu'il existe un risque que l'environnement Web soit usurpé par un certificat frauduleux ou non légitime. Pour toutes les mesures de protection suggérées dans ce rapport, il est recommandé d'évaluer le bénéfice de sécurité obtenu par rapport à la difficulté ou à la complexité de la mise en œuvre et de la gestion[47]. Cependant, HPKP peut être mis en œuvre sans aucun risque en utilisant uniquement ses capacités de notification (décrites ci-dessous), ce qui permet d'utiliser les navigateurs et clients Web comme capteurs pour détecter des attaques par usurpation d'identité en cas d'identification de certificats numériques non légitimes.

HPKP est probablement l'un des mécanismes de sécurité les plus stricts disponibles de nos jours, et il introduit le risque de bloquer l'accès à l'environnement Web s'il n'est pas configuré correctement

L'objectif de HPKP est d'atténuer le risque associé à l'une des principales faiblesses de l'écosystème mondial des clés publiques (PKI) de l'Internet : le fait que n'importe quelle AC publique (il en existe actuellement des centaines) puisse générer un certificat numérique valide pour n'importe quel environnement Web de l'Internet, et ce sans l'autorisation expresse de son propriétaire. HPKP empêche, dans le client Web lui-même, les attaques MitM (Man-in-the-Middle)[48] qui utilisent des

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

[48] Principalement en raison de la compromission d'une AC, ou d'un comportement malveillant ou anormal de la part d'une AC.

certificats valides mais qui ne sont pas légitimes ou autorisés (c'est-à-dire qui ont été émis par l'une des AC reconnues publiquement, bien que leur émission n'ait pas été autorisée par le propriétaire du serveur Web). En fait, HPKP permet de gérer (supprimer) la confiance dans les AC et les PKI par défaut, et de ne faire confiance qu'à un ensemble très spécifique de certificats numériques (épingles).

Cette mesure de sécurité a été conçue à la suite de la multiplication des incidents de sécurité impliquant des AC au long de ces dernières années, avec des cas bien connus comme DigiNotar ou Comodo en 2011, ou StartCom et WoSign en 2016 (avec de multiples violations des règles du jeu des AC), et qui ont permis aux attaquants (ou aux AC) de générer des certificats numériques pour de nombreux sites Web non légitimes, mais parfaitement valables pour tous les navigateurs et clients Web.

L'en-tête HPKP doit être inclus dans chaque réponse HTTPS générée par le serveur ou l'application Web :

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

REMARQUE :
Dans le cas de HPKP, comme pour CSP, il est possible d'appliquer strictement et efficacement la politique tout en recevant des rapports de violation (via "report-uri"). Cette option est très intéressante pour identifier les attaques utilisant des certificats non légitimes.

Il convient de noter que le navigateur Web impose l'utilisation de HTTPS pour un ensemble spécifique de certificats numériques pendant la période indiquée par la directive "max-age" (spécifiée en secondes) de l'en-tête HPKP. Il est recommandé d'utiliser une valeur "max-age" de 5184000 (60 jours ou 2 mois), comme le recommandent les RFC, afin de trouver un équilibre entre sécurité et disponibilité.

Pour minimiser l'impact éventuel d'une configuration incorrecte lors du démarrage du déploiement de HPKP, il est recommandé d'augmenter progressivement la valeur "max-age", comme cela est suggéré pour l'en-tête HSTS (voir section "3.7.4. Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS"). En même temps, les accès à l'URL de rapport indiquée par la directive "report-uri" doivent être surveillés de près.

La directive "report-uri" fait référence à l'adresse Web où les navigateurs et clients doivent signaler les erreurs de validation des épingles, à la fois en mode standard et en mode rapport HPKP, en utilisant le format défini par les RFC. Il est recommandé d'utiliser une référence Web HTTPS associée à un serveur Web autre que celui qui a envoyé l'en-tête HPKP, sinon il ne sera pas possible pour le navigateur Web de soumettre le rapport en raison de l'erreur d'épinglage. Pour recevoir ces rapports, il est possible d'utiliser votre propre serveur Web spécifique, sur le même domaine ou sur un autre, ou d'utiliser un service tiers tel que "Report URI"[49] [50].

La directive "report-uri" fait référence à l'adresse Web où les navigateurs et clients doivent signaler les erreurs de validation des épingles, à la fois en mode standard et en mode rapport HPKP, en utilisant le format défini par les RFC

Du point de vue de la sécurité, dans le cas de la spécification d'épingles associées à des certificats de type *wildcard* ou à des AC racine ou intermédiaires, pour des raisons de simplicité, HPKP doit être mis en œuvre pour l'ensemble du domaine, et pas seulement pour des serveurs Web individuels. Pour ce faire, la directive "includeSubDomains" doit être utilisée.

De plus, comme pour les HSTS (voir section "3.7.4. Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS"), il est important de noter que HPKP est un mécanisme de sécurité TOFU (Trust On First Use), c'est-à-dire qu'il ne s'applique pas la première fois que le client Web se connecte au serveur Web. Dans le cas du HPKP, cependant, il n'existe pas de liste préchargée accessible au public comme c'est le cas pour le HSTS. En interne, les navigateurs Web disposent d'une liste ou d'un ensemble statique d'épingles (static_spki_hashes), appelé "Static Public Key Pinning", qui s'applique à certains domaines associés aux entreprises liées aux navigateurs Web en soi (Google, Firefox, etc.) ou, exceptionnellement, à d'autres grandes entreprises et domaines inclus par celles-ci (Facebook, Twitter, etc.)[51].

De plus, comme pour les autres en-têtes HTTP de sécurité tels que CSP (voir section "3.7.6. *Content Security Policy* (CSP)"), et pour minimiser l'impact potentiel d'une configuration incorrecte lors du démarrage du déploiement de HPKP, il est recommandé de commencer par utiliser les capacités de rapport de HPKP via l'en-tête étendu "...-Report-Only" et la directive "report-uri". Cet en-tête doit être testé initialement pour vérifier le bon fonctionnement de la norme. Si un navigateur ou un client Web, capable de prendre en charge HPKP Report-Only[52] (comme Chrome 46

[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

ou Opera 33[53]), reçoit un certificat numérique non légitime, c'est-à-dire non inclus dans les épingles de l'en-tête HPKP, il ne bloquera pas le trafic HTTPS (comme c'est le cas avec l'en-tête HPKP standard, qui applique la politique HPKP sans exception), mais générera un rapport dirigé vers l'URL indiquée dans l'en-tête par la directive "report-uri". Dans ce scénario, les clients Web sont effectivement utilisés comme des capteurs pour détecter les attaques par usurpation d'identité sur le Web :

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

REMARQUE :
En cas d'utilisation de "...-Report-Only", il n'est pas nécessaire d'utiliser la directive "max-age".

Il est recommandé de commencer à utiliser l'en-tête HPKP uniquement sur une ressource Web spécifique, jusqu'à ce que sa configuration correcte soit vérifiée, afin d'éviter de recevoir une multitude de rapports de violation ou d'erreurs de validation d'épingles.

Deux éléments clés à prendre en compte lors de la mise en œuvre de HPKP (décrits ci-dessous) sont la manière de générer les épingles pour les certificats numériques existants, et sur quel certificat numérique de la chaîne de certification actuelle l'épingle reproduite dans l'en-tête HPKP doit être associée.

Lors de la création d'épingles ou de références à des certificats numériques légitimes, il existe théoriquement deux options : se référer directement au certificat numérique ou à la clé publique incluse dans le certificat numérique. HPKP fait référence à la clé publique, plus précisément au champ "Subject Public Key Info" (SPKI) des certificats numériques X.509, pour éviter de dépendre du renouvellement et des autres tâches de gestion des certificats numériques. La structure SPKI comprend la clé publique ainsi que certaines métadonnées, telles que l'algorithme cryptographique utilisé pour sa génération (p. ex. RSA).

La vérification HPKP dans les navigateurs ou clients Web exige donc que la chaîne de certification réelle présentée au navigateur ou au client Web par l'environnement, le serveur ou l'application Web comprenne au moins une structure SPKI dont l'empreinte digitale (*fingerprint*) correspond à l'empreinte digitale (*fingerprint*) de l'une des épingles de l'en-tête HPKP.

[53] Pas encore pris en charge dans Firefox : https://bugzilla.mozilla.org/show_bug.cgi?id=1091177

L'en-tête HPKP doit comprendre, au moins, deux épingles (représentées ci-dessus par A et B) : une épingle primaire liée à la valeur SPKI de l'un des certificats numériques de la chaîne de certification actuelle, et une épingle de *secours* (*backup pin*), qui ne doit pas faire partie de la chaîne de certification actuelle. L'épingle de secours peut être utilisée, par exemple, pour renouveler la clé cryptographique (publique et privée) du serveur et son certificat numérique associé, après un incident de sécurité où l'épingle principale est compromise. En outre, il est possible de configurer davantage d'épingles dans l'en-tête HPKP, qu'il s'agisse d'épingles principales ou de secours.

La valeur représentée par l'en-tête HPKP dans la directive "pin-sha256", associée à la valeur de l'épingle, correspond à un *hachage* SHA-256 de la valeur du champ SPKI en base64. La présentation "HTTPS : TLS, not SSL" [Réf. - 2] fournit les détails nécessaires pour générer les épingles (primaire et de secours) en utilisant OpenSSL.

Du point de vue de la gestion et du renouvellement des clés cryptographiques (voir section "3.2. Génération des clés cryptographiques"), il convient de noter qu'il est courant de renouveler les clés cryptographiques à chaque fois que le certificat numérique associé est renouvelé (voir section "3.3.5. Renouvellement des certificats numériques", comme c'est le cas par défaut pour l'AC Let's Encrypt). Cependant, comme les épingles HPKP sont associées à la clé cryptographique (ou clé publique), il serait nécessaire de les mettre à jour si la clé est renouvelée. Dans ce cas, il est recommandé de ne pas renouveler la clé cryptographique au cours du processus standard de renouvellement des certificats numériques, afin de maintenir la valeur des épingles existantes.

Lors de l'association de l'épingle représentée dans l'en-tête HPKP à un certificat numérique de la chaîne de certification actuelle, il existe plusieurs possibilités, la plus restrictive étant celle qui préfère créer une épingle pour la clé publique (et, par conséquent, le certificat numérique final) du serveur Web, et la plus permissive celle qui associe l'épingle à la clé publique de l'AC racine. Il est également possible d'associer l'épingle à la clé publique de l'une des AC intermédiaires existant dans la chaîne de certification actuelle. Il faut trouver un équilibre entre le niveau de sécurité (ou de restriction) à établir par HPKP, les risques éventuels d'accessibilité et les tâches de gestion associées aux clés cryptographiques. Les considérations à prendre en compte s'appliquent à la fois aux épingles principales et aux épingles de secours.

L'en-tête HPKP doit comprendre, au moins, deux épingles : une épingle primaire liée à la valeur SPKI de l'un des certificats numériques de la chaîne de certification actuelle, et une épingle de secours, qui ne doit pas faire partie de la chaîne de certification actuelle

Le scénario le plus sûr et le plus restrictif est celui qui associe l'épingle à la clé publique du serveur Web lui-même. Toutefois, il faut veiller à modifier les épingles de manière appropriée, en anticipant le changement et/ou le renouvellement de la clé, et à inclure des épingles pour toutes les clés associées au serveur Web (p. ex. si l'on utilise simultanément des clés RSA et ECDSA ; voir section "3.2. Génération des clés cryptographiques").

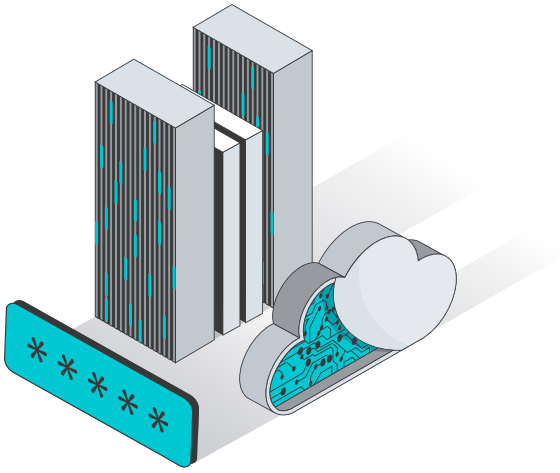
Le scénario le plus sûr et le plus restrictif est celui qui associe l'épingle à la clé publique du serveur Web lui-même

Le scénario le plus souple et le plus permissif est celui qui associe l'épingle à la clé publique d'une AC intermédiaire, ou plus encore, de l'AC racine. Cependant, il doit également être géré avec soin afin de conserver les épingles actuelles (si on ne souhaite pas les changer), notamment lors du processus de renouvellement du certificat numérique du serveur Web, car les AC disposent généralement de plusieurs AC racines et intermédiaires. Il est donc important de s'assurer que le nouveau certificat numérique sera émis par la même AC. En outre, les AC établissent des relations de confiance entre elles en signant certains certificats numériques de façon croisée, ce qui signifie qu'il peut y avoir plusieurs chaînes de certificats valides pour le même certificat numérique. Dans ce cas, il est nécessaire de disposer d'épingles pour toutes les chaînes de certification valides possibles.

En essayant de trouver un équilibre entre les deux scénarios et les avantages de chacun, une option consiste à créer l'épingle pour la première AC intermédiaire, c'est-à-dire celle qui a directement émis le certificat numérique actuel du serveur Web.

D'autre part, l'épingle (ou les épingles) de secours peut (peuvent) être associée(s) à la même AC ou, mieux encore, à une AC différente, au cas où l'AC actuelle serait compromise. Il est également recommandé de conserver la clé privée et le certificat numérique associés à l'épingle de sauvegarde hors ligne, c'est-à-dire dans un endroit sûr où il est plus difficile qu'elles soient compromises.

Enfin, il convient de noter que HPKP n'offre pas de protection contre la manipulation locale des AC dans les navigateurs et clients Web, comme lorsqu'un utilisateur importe une nouvelle AC racine. Les certificats numériques associés à cette AC racine seront considérés comme valides, qu'ils correspondent ou non aux épingles de l'en-tête HPKP, un scénario qui permet d'utiliser des proxies Web locaux et des solutions d'entreprise pour l'inspection et l'interception du trafic HTTPS (voir section "3.7.9. Inspection du trafic HTTPS").



3.3.10 Recommandations pour la gestion des certificats numériques

Le résumé des recommandations pour la gestion des certificats numériques, où de multiples aspects doivent être pris en compte, comprend :

- Pour les serveurs Web exposés publiquement, il est nécessaire de disposer d'un certificat numérique émis par une autorité de certification publique fiable et largement reconnue. Cependant, les certificats numériques auto-signés et les AC privées ne sont pas valables.
- Évaluez et sélectionnez le type approprié de certificat numérique (DV, OV ou EV) à obtenir pour chaque serveur ou application Web.
- Évaluez et sélectionnez l'AC qui délivrera le certificat en fonction de multiples critères objectifs, tels que : sa réputation, les fonctionnalités qu'elle offre, son adoption de nouvelles normes, les mécanismes de révocation de certificats qu'elle met en œuvre, etc.
- Sélectionnez la période de renouvellement du certificat numérique et renouvelez les certificats périodiquement et toujours avant leur expiration.
- Utilisez des certificats numériques signés en utilisant SHA-256 (ou plus). N'utilisez en aucun cas des signatures basées sur SHA-1 ou MD5.
- Identifiez et configurez correctement (dans le bon ordre) la chaîne de certification (ou de certificats) complète associée au certificat numérique, de l'AC racine au certificat final en passant par toutes les AC intermédiaires.
- Sélectionnez le type de certificat numérique approprié en fonction du nom de l'entité qu'il représente, c'est-à-dire : référence directe ou unique, liste de noms, certificat *Wildcard* ou multi-domaine.
- Le certificat numérique doit inclure à la fois le nom du serveur Web représenté ("www", ou la liste des serveurs Web, ou le *Wildcard*) et son domaine.
- Le champ "SAN" (*Subject Alternative Name*, extension "subjectAltName") du certificat numérique doit spécifier le nom de l'entité, et non pas le "CN" (*Common Name*).

- Garantissez la validité du certificat numérique à tout moment, y compris le nom de l'entité, la période de validité, l'AC émettrice du certificat, la révocation du certificat et son objet.
- Associez le certificat numérique à une AC possédant la qualification de transparence (CT) et confirmez que le certificat a bien été enregistré dans les journaux CT publics.
- Surveillez les *logs* ou les journaux CT pour identifier les certificats numériques émis de manière illégitime par d'autres AC, sans l'autorisation du propriétaire du domaine. En complément, utilisez les services de surveillance des CT pour être informé des changements dans les certificats numériques associés à un domaine.
- Fournissez les journaux CT (via SCT) dans les réponses HTTPS du serveur Web aux clients Web via une extension de certificat numérique X.509v3, ou (de préférence) via une extension TLS spécifique ou encore en utilisant l'*OCSP Stapling*.
- Il est recommandé d'envisager l'utilisation de l'en-tête HTTP "*Expect-CT*" pour indiquer aux sites de choisir de signaler et/ou d'appliquer les exigences de CT. Dans le cas contraire, utilisez les capacités de notification de "*Expect-CT*" pour identifier l'utilisation inattendue de certificats mal émis pour ce site.
- Configurez dans le service de résolution de noms DNS les enregistrements CAA correspondant à la liste des AC associées aux certificats numériques du domaine.
- Associez le certificat numérique à une AC utilisant la CAA (*Certification Authority Authorization*).
- Utilisez les capacités de notification de la CAA pour identifier les erreurs ou les demandes de certificats invalides.
- N'utilisez en aucun cas HPKP (*HTTP Public Key Pinning*).
- Sélectionnez l'AC liée aux épingles de sauvegarde HPKP et protégez la clé privée associée à ces épingles en conséquence.

3.4. Protocoles SSL et TLS

Cette section aborde toutes les variantes et les fonctionnalités supplémentaires associées au protocole obsolète SSL (*Secure Sockets Layer*) et à son remplaçant, le protocole TLS (*Transport Layer Security*).

Cette section aborde toutes les variantes et les fonctionnalités supplémentaires associées au protocole obsolète SSL et à son remplaçant, le protocole TLS

3.4.1 Versions des protocoles SSL et TLS

La sélection des protocoles SSL et TLS, et des versions spécifiques de ces protocoles, à utiliser par le serveur ou l'application Web sera influencée par les exigences de sécurité et d'interopérabilité de l'environnement Web, c'est-à-dire quels sont les navigateurs et clients Web qui doivent être pris en charge et auxquels on veut donner accès avec le plus haut niveau de sécurité possible.

Les différentes versions du protocole TLS ont apporté des améliorations en matière de sécurité. Par exemple, TLS 1.0 a ajouté l'utilisation de fonctions pseudo-aléatoires (PRF, *Pseudo-Random Function*) basées sur HMAC, les PRF étant également utilisées pour générer le secret principal. Les capacités de *padding* ont également été améliorées dans TLS 1.0 par rapport à SSL 3.0. La version TLS 1.1 a ajouté principalement des améliorations de sécurité par rapport à TLS 1.0 —telles que l'utilisation du mode de chiffrement CBC avec des vecteurs d'initialisation (IV) explicites dans chaque enregistrement TLS au lieu d'IV prévisibles—, ainsi que des améliorations de *padding*, et a inclus des extensions TLS (RFC 3546[54]). La version TLS 1.2 a ajouté le chiffrement authentifié (AEAD), la prise en charge de HMAC-SHA256 (utilisé pour la fonction PRF standard et donc pour les algorithmes de signature, bien qu'il soit possible d'utiliser d'autres fonctions PRF), une extension permettant aux deux extrémités d'indiquer et de négocier les algorithmes de signature et de *hachage* qu'elles prennent en charge, et a éliminé l'utilisation d'algorithmes cryptographiques faibles.

[54] https://tools.ietf.org/html/rfc3546

La version TLS1.3 apporte d'autres améliorations, notamment la suppression des chiffrements et des algorithmes faibles (seuls cinq sont désormais recommandés), la simplification de l'échange de clés (et l'amélioration de la vitesse d'échange de clés) et l'amélioration des performances pour les sites déjà visités grâce à des mécanismes tels que le *handshake* "zero round trip time (0-RTT)".

L'utilisation du protocole SSL est déconseillée, car il est considéré comme totalement vulnérable depuis sa version 2.0, en raison de vulnérabilités antérieures, mais surtout après la découverte de la vulnérabilité DROWN en mars 2016, jusqu'à sa dernière version 3.0, après la découverte de la vulnérabilité POODLE en octobre 2014 (voir section "3.8. Vulnérabilités dans HTTPS").

Par conséquent, dans le cas le plus tolérant du point de vue de l'interopérabilité, l'environnement Web ne devrait supporter que les différentes versions du protocole TLS. Toutefois, en raison des faiblesses existantes dans la version 1.0 de ce protocole, comme BEAST (entre autres), publié en juin 2011 (voir section "3.8. Vulnérabilités dans HTTPS"), il est recommandé de trouver un équilibre entre sécurité et interopérabilité et de ne prendre en charge que TLS 1.1 et TLS 1.2 (RFC 5246[55]), ainsi que les futures versions du protocole, telles que TLS 1.3 [Réf. - 22][56].

À titre de référence, tous les sites Web qui acceptent les paiements par carte de crédit et qui sont donc soumis à la réglementation PCI (*Payment Card Industry*) doivent supprimer la prise en charge de TLS 1.0 d'ici juillet 2018, date initialement prévue pour juillet 2016[57] (indiquant l'obsolescence de cette version du protocole, qui a été publiée en janvier 1999[58]).

Depuis le 25 mars 2021, date de la dernière mise à jour de la norme RFC 8996, les protocoles TLS 1.0 et 1.1 sont obsolètes.

TLS 1.2 et 1.3 sont les seules versions qui prennent en charge les algorithmes cryptographiques modernes, tels que ceux associés au chiffrement authentifié (*Authenticated Encryption*, AE), ou AEAD (voir section "3.5.5. Robustesse des algorithmes et des suites cryptographiques").

La version 1.3 du protocole TLS est déjà prise en charge par tous les principaux navigateurs à partir des versions suivantes :

- **Google Chrome, version 67 et ultérieure**
- **Mozilla Firefox, version 61 et ultérieure**
- **Mac OS 10.3, iOS 11 et ultérieure**

[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

3.4.2 SNI (*Server Name Indication*)

SNI (*Server Name Indication*) [Réf. - 8] est une extension du protocole TLS qui permet aux clients Web d'indiquer le nom du serveur Web (nom d'hôte) auquel ils souhaitent se connecter pendant le processus d'établissement de la connexion HTTPS initiale à l'aide du protocole TLS (*handshake*).

Les extensions TLS sont un mécanisme disponible pour étendre la fonctionnalité de TLS sans avoir à modifier le protocole. Les extensions ont des implications positives pertinentes du point de vue de la sécurité, et sont utilisées pour des fonctionnalités telles que SNI, l'*OCSP Stapling* ("3.6.3. *OCSP Stapling*"), les tickets de session ("3.7.2.2. *TLS session resumption* ou *TLS session caching*, et *TLS session tickets*"), ou des mécanismes de renégociation sécurisés (" 3.4.3. Renégociation"), entre autres.

En utilisant SNI, il est possible d'utiliser HTTPS dans des environnements avec des serveurs Web ou des *hôtes* virtuels (HTTP/1.1), c'est-à-dire lorsque différents serveurs Web (noms d'hôtes) coexistent sur le même serveur physique et utilisent la même adresse IP. SNI permet donc d'avoir des certificats numériques différents pour chacun des serveurs Web, tous fournis à partir de la même adresse IP et du même port HTTPS (TCP/443).

Si le support SNI n'est pas disponible, il n'est possible d'utiliser qu'un seul serveur Web (ou un très petit groupe de serveurs) par adresse IP, car un seul certificat numérique lui est associé. Ce certificat numérique pourrait être partagé entre plusieurs serveurs Web associés à cette adresse IP, mais cette option n'est recommandée que s'il s'agit d'un certificat numérique (p. ex. Wildcard) qui est valable pour tous les différents serveurs Web du même domaine disponibles à cette adresse IP.

Étant donné que SNI n'est pas pris en charge sur les anciennes versions d'Android <= 2.3.x, de Java <= 6 et d'Internet Explorer sur Windows XP (dont la prise en charge s'est terminée en 2014), du point de vue de l'interopérabilité, il est acceptable de l'utiliser.

SNI est une extension du protocole TLS qui permet aux clients Web d'indiquer le nom du serveur Web (nom d'hôte) auquel ils souhaitent se connecter pendant le processus d'établissement de la connexion HTTPS initiale à l'aide du protocole TLS (*handshake*)

Le seul inconvénient de SNI est que le nom du serveur Web est révélé en clair par le client Web dans le trafic réseau avant que le tunnel chiffré ne soit établi via HTTPS, bien que dans la plupart des scénarios, ce ne soit pas un inconvénient, car le nom du serveur Web est soit connu publiquement, soit révélé via le service DNS.

Dans le cas de plusieurs serveurs Web ou hôtes virtuels liés à la même adresse IP, il est recommandé de prendre en charge SNI dans l'implémentation de HTTPS.

3.4.3 Renégociation

L'implémentation du protocole TLS fournit des mécanismes de renégociation, qui permettent au client et au serveur de renégocier les détails et les paramètres d'une session TLS déjà établie, par exemple lors de l'utilisation de certificats numériques du client, tels que ceux associés aux cartes d'identité électroniques (DNIe) ou aux certificats d'utilisateur.

L'implémentation du protocole TLS fournit des mécanismes de renégociation, qui permettent au client et au serveur de renégocier les détails et les paramètres d'une session TLS déjà établie

Les mécanismes de renégociation TLS n'étaient pas sûrs, facilitant l'interception des communications (MitM) et les attaques par déni de service (DoS, *Denial of Service*) sur le serveur Web, car l'opération de renégociation nécessite plus de ressources sur le serveur que sur le client.

Il est recommandé d'utiliser une implémentation TLS faisant appel aux mécanismes de renégociation sécurisée définis dans la norme RFC 5746[59] et, de préférence, que la renégociation soit toujours initiée par le serveur. Si les certificats numériques des clients ne sont pas utilisés, il est recommandé de désactiver complètement les mécanismes de renégociation. Au minimum, les mécanismes qui permettent au client d'initier la renégociation doivent être désactivés. Par exemple, Apache ne prend plus en charge la renégociation initiée par le client depuis la version 2.2.15.

[59] https://tools.ietf.org/html/rfc5746

3.4.4 Dégradation de la version du protocole TLS

L'une des principales faiblesses associées au protocole HTTPS est la possibilité que, pendant le processus de négociation associé à l'établissement de la session chiffrée, un attaquant potentiel puisse forcer le client et/ou le serveur à utiliser une version antérieure (*downgrade*) et vulnérable des protocoles SSL ou TLS. Cette technique a été utilisée, par exemple, dans l'attaque POODLE pour dégrader la connexion à la version 3.0 de SSL, même à partir de versions plus sûres du protocole, comme TLS 1.2.

Pour éviter de tels scénarios d'attaque, il est recommandé d'implémenter (dans le serveur Web) la *TLS Fallback Signaling Cipher Suite Value* (SCSV), RFC 7507[60], une extension du protocole TLS qui empêche les attaques par dégradation du protocole. Cette extension est prise en charge par Chrome 33[61] et Firefox 35[62], et par OpenSSL version 1.0.1j (et ultérieures).

3.4.5 HTTP/2

Actuellement, l'impact le plus important sur les performances de TLS (voir section "3.7.2. Performances") n'est généralement pas associé à des opérations cryptographiques coûteuses du point de vue de la CPU, mais à la latence du réseau.

Actuellement, l'impact le plus important sur les performances de TLS n'est généralement pas associé à des opérations cryptographiques coûteuses du point de vue de la CPU, mais à la latence du réseau

Pour cette raison, et pour améliorer les performances de l'environnement Web, il est recommandé d'établir le nombre minimum possible de connexions HTTPS (et donc TCP). Cela peut se faire en augmentant la durée des sessions TCP (à l'aide du paramètre *TCP keep alive*) ou en utilisant HTTP/2, un protocole qui maintient une seule session TCP sur le serveur Web pour chaque client Web. HTTP/2 est actuellement pris en charge par la plupart des navigateurs Web modernes[63].

En outre, la distance entre le client et le serveur Web est un facteur critique en termes de latence de communication. Cette distance peut être optimisée par l'utilisation de CDN (*Content Delivery Networks*) dotés de capacités HTTPS, bien qu'il soit recommandé d'évaluer les détails de l'implémentation HTTPS de ce type de services, et les différentes modalités offertes par ceux-ci[64].

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/

3.4.6 Recommandations relatives aux protocoles SSL et TLS

Voici, ci-dessous, le résumé des recommandations relatives aux protocoles SSL et TLS :

- Désactivez complètement la prise en charge de SSL 2.0 et SSL 3.0.
- Désactivez complètement la prise en charge du protocole TLS 1.0 et TLS 1.1.
- Utilisez uniquement les versions 1.1 et 1.2 du protocole TLS.
- Effectuez les actions nécessaires pour supporter TLS 1.3.
- Garantissez un support pour SNI dans la mise en œuvre de HTTPS dans le cas de plusieurs serveurs Web ou hôtes virtuels liés à la même adresse IP.
- Désactivez les mécanismes de renégociation TLS, en particulier la renégociation qui peut être initiée par le client. Si ces capacités sont nécessaires, activez la renégociation initiée par le serveur et utilisez les mécanismes de renégociation TLS sécurisés.
- Activez l'extension du protocole TLS "*TLS Fallback Signaling Cipher Suite Value*" (SCSV) pour empêcher les attaques par dégradation (*downgrade*) du protocole TLS.
- Évaluez la possibilité de mettre en œuvre HTTP/2 sur le serveur Web, en coexistant avec la version HTTP/1.1 du protocole, en raison des avantages en termes de rendement de l'utilisation de HTTPS.

3.5. Algorithmes et suites cryptographiques

La configuration des algorithmes et des suites cryptographiques pris en charge par l'environnement Web basé sur le protocole HTTPS est l'un des éléments les plus complexes à prendre en compte dans la mise en œuvre du protocole HTTPS, car elle doit toujours rechercher un équilibre entre le niveau de sécurité fourni (basé sur la forteresse des algorithmes et des protocoles cryptographiques utilisés, ainsi que sur les développements industriels et les attentes à moyen et long terme de ceux-ci), les performances de l'environnement Web et l'interopérabilité avec plusieurs navigateurs et clients Web.

La configuration des algorithmes et des suites cryptographiques pris en charge par l'environnement Web basé sur le protocole HTTPS est l'un des éléments les plus complexes à prendre en compte dans la mise en œuvre du protocole HTTPS, car elle doit toujours rechercher un équilibre entre le niveau de sécurité fourni, les performances de l'environnement Web et l'interopérabilité avec plusieurs navigateurs et clients Web

3.5.1 *Forward Secrecy*

L'échange de clés basé sur le RSA ne fournit pas de *Forward Secrecy* (également appelé PFS, *Perfect Forward Secrecy*, ou Confidentialité persistante), c'est-à-dire une propriété cryptographique dont le but est d'empêcher le trafic chiffré capturé dans le passé d'être déchiffré plus tard si les clés sont compromises dans le futur. En d'autres termes, la compromission ou l'obtention des clés de chiffrement actuelles ne permet pas de compromettre ou d'obtenir les clés de chiffrement utilisées dans le passé. Fondamentalement, le *Forward Secrecy* empêche le trafic chiffré de s'appuyer sur la clé privée. Il est préférable d'utiliser des algorithmes basés sur le protocole Diffie-Hellman (DH), tels que DHE (*DH Ephemeral*, ou EDH) ou ECDHE (*Elliptic Curve DHE*, ou EECDH), qui utilisent des paramètres DH éphémères (et produisent des clés éphémères). Du point de vue de la sécurité et des performances, ECDHE est préférable à DHE.

Le *Forward Secrecy* garantit que chaque session TLS de chaque client (aussi vers le même serveur) utilise des clés cryptographiques différentes, alors que sans celui-ci (p. ex. en utilisant RSA), les clés de toutes les sessions dépendent de la clé privée du serveur Web et pourraient être déchiffrées à partir de celle-ci.

Toutefois, le RSA peut toujours être utilisé pour générer les clés associées aux certificats numériques. Il est important de faire la distinction entre les algorithmes et les clés cryptographiques utilisés pour l'identification des terminaux et la génération des certificats numériques associés (voir section "3.2. Génération des clés cryptographiques"), et les algorithmes utilisés pour l'échange de clés lors de l'établissement d'une session TLS (voir section "3.5.2. Échange de clés robuste"), où l'utilisation du *Forward Secrecy* doit être garantie.

3.5.2 Échange de clés robuste

La section "3.5.1. *Forward Secrecy*" décrit en détail la principale propriété requise lors de l'échange de clés lors de l'établissement d'une session TLS : le *Forward Secrecy*. À cette fin, il convient d'éviter les suites cryptographiques qui utilisent l'échange de clés basé sur RSA, et d'employer plutôt la variante à courbe elliptique (ECDHE) ou, alternativement, la variante classique d'échange de clés Diffie-Hellman (DHE) qui génère des clés éphémères (*ephemeral*). En réalité, RSA est utilisé pour transporter la clé maîtresse (ou de session), tandis que (EC)DHE est utilisé pour négocier la clé maîtresse, en utilisant le *Forward Secrecy*.

Trois algorithmes d'échange de clés sont donc disponibles (dans l'ordre inverse de priorité) : RSA, DHE et ECDHE. En résumé, l'échange de clés via ECDHE devrait être effectué avec tous les navigateurs et clients Web modernes et, alternativement (pour des raisons de compatibilité), DHE devrait être pris en charge pour utiliser le *Forward Secrecy* avec les navigateurs et clients Web plus anciens (tels que Android < 3.0, Java < 7 ou OpenSSL < 1.0.0).

La nomenclature utilisée pour les algorithmes d'échange de clés (voir section "3.5.5. Robustesse des algorithmes et des suites cryptographiques") comprend l'algorithme d'échange de clés (RSA, DHE et ECDHE) et la méthode d'authentification (basée sur le type de clés utilisées pour générer les certificats numériques : RSA ou ECDSA). Les combinaisons les plus courantes, par ordre inverse de priorité, sont les suivantes :

- **RSA :** échange de clés RSA avec authentification RSA.
- **DHE_RSA :** Échange de clés DH éphémères avec authentification RSA.
- **ECDHE_RSA :** Échange de clés ECDH éphémères avec authentification RSA.
- **ECDHE_ECDSA :** Échange de clés ECDH éphémères avec authentification ECDSA.

REMARQUE :
En outre, des suites cryptographiques utilisant la cryptographie post-quantique (CECPQ1) pour l'échange de clés[65], combinant l'algorithme NewHope avec des courbes elliptiques X25519, sont devenues disponibles dans TLS récemment.

Après l'échange de clés entre le client et le serveur, quelle que soit la méthode utilisée, les deux extrémités disposent d'un secret pré-maître (*premaster secret* ou *premaster key*). Après avoir appliqué l'algorithme d'échange de clés sélectionné, et vu que chaque algorithme peut gérer différentes longueurs de clés, la valeur de sortie est traitée par une fonction pseudo-aléatoire (PRF) pour obtenir toujours un secret maître de 48 octets (384 bits).

Dans TLS, l'échange de clés est lié au processus d'authentification, la clé publique (RSA ou ECDSA) étant utilisée à la fois pour vérifier que la communication est établie avec l'entité souhaitée et pour dériver le secret principal à l'aide de l'algorithme d'échange de clés choisi. Dans le cas de RSA, le client génère le secret maître et envoie celui-ci chiffré avec la clé publique du serveur ; seul le serveur peut le déchiffrer avec la clé privée correspondante, s'authentifiant ainsi implicitement. Dans le cas de DHE ou ECDHE, les paramètres DH envoyés par le serveur sont signés avec sa clé privée, afin que le client puisse vérifier la signature avec la clé publique correspondante et confirmer qu'il communique avec le serveur attendu (authentification).

[65] https://www.imperialviolet.org/2016/11/28/cecpq1.html

En règle générale, il est recommandé d'utiliser des algorithmes d'échange de clés de 256 bits pour ECDHE et de 2.048 bits pour DHE (voir section "3.5.3. Robustesse des paramètres Diffie-Hellman (DH)").

Il est donc recommandé d'employer ECDHE en utilisant, par exemple, la courbe "secp256r1" (appelée "prime256v1" dans OpenSSL ou P-256, et largement supportée par les clients Web), qui fournit une sécurité de 128 bits pour l'échange de clés. Une autre solution consiste à utiliser la courbe "secp384r1" (connue sous le nom de P-384 et supportée généralement, RFC 4492[66]), mais ses performances sont inférieures à celles de la courbe P-256 et elle n'offre pas une amélioration de la sécurité suffisamment significative pour justifier son utilisation. On peut aussi utiliser la courbe "secp521r1", mais sa prise en charge est plus limitée.

Ces deux premières courbes, définies par le NIST, doivent être utilisées avec prudence, car il n'est pas clair comment leurs paramètres ont été sélectionnés et elles peuvent donc présenter des faiblesses que seuls leurs concepteurs connaissent. Par ailleurs, deux courbes standard connues sous le nom de Curve25519 (X2559) et Curve448 (X448) sont devenues populaires récemment, même pour être utilisées dans TLS[67] [68].

Alternativement, pour les cas où il n'est pas possible pour le client Web d'utiliser ECDHE, il est recommandé de fournir un support pour DHE en utilisant des paramètres robustes de 2.048 bits (voir section "3.5.3. Robustesse du paramètre Diffie-Hellman (DH)").

En règle générale, il est recommandé d'utiliser des algorithmes d'échange de clés de 256 bits pour ECDHE et de 2.048 bits pour DHE

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

3.5.3 Robustesse des paramètres Diffie-Hellman (DH)

La longueur ou la taille des paramètres Diffie-Hellman (DH) doit être d'au moins 2.048 bits et il est recommandé qu'elle corresponde à la longueur ou à la forteresse de la clé privée correspondante.

Parmi les incompatibilités connues des clients Web, nous pouvons citer le cas de Java 6, qui ne prend en charge que les paramètres DH de 1.024 bits. C'est également le cas de Java 7, mais sans conflit du point de vue de la sécurité, car il prend également en charge ECDHE.

Dans tous les cas, il n'est pas recommandé d'utiliser des groupes DH connus ou préconfigurés (de 1.024 bits), tels que ceux recommandés dans la norme RFC 3526[69], ou des paramètres DH de 768 bits ou moins, surtout après la découverte de la vulnérabilité Logjam en janvier 2015 (voir section "3.8. Vulnérabilités dans HTTPS"). De même, les implémentations HTTPS doivent être très prudentes dans la sélection des paramètres DH, car, comme l'a démontré la vulnérabilité du triple *handshake* en 2014 (voir section "3.8. Vulnérabilités dans HTTPS"), il était même possible pour un serveur malveillant de sélectionner des paramètres DH non sécurisés qui n'étaient même pas des nombres premiers.

Jusqu'à l'introduction de la norme RFC 7919[70] (août 2016), les clients ne pouvaient pas spécifier leurs exigences de sécurité concernant les paramètres DH pendant le processus d'établissement et de négociation de la session TLS, et ils devaient accepter ceux fournis par le serveur. Sur la base de la norme RFC 7919, il est recommandé, au lieu d'utiliser des groupes DH préconfigurés ou de générer vos propres groupes avec "openssl dhparam", d'utiliser les groupes DH prédéfinis dans cette norme, car ils ont été vérifiés du point de vue de la sécurité : ffdhe2048, ffdhe3072 ou ffdhe4096[71].

[69] https://tools.ietf.org/html/rfc3526

[70] https://tools.ietf.org/html/rfc7919

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

3.5.4 Les préférences entre suites cryptographiques

L'une des fonctionnalités fondamentales du protocole TLS est la capacité du client et du serveur HTTPS à négocier, pendant l'établissement de la session TLS (ou *handshake* TLS), l'ensemble des suites cryptographiques qui pourraient être utilisées par les deux parties, et à parvenir à un accord ou un consensus sur la meilleure option disponible en fonction des priorités ou des préférences des deux extrémités. Cette négociation permet au serveur ou à l'application Web de toujours essayer d'utiliser l'option la plus sûre disponible avec chacun des clients Web qui s'y connectent.

Il est donc recommandé de configurer le serveur Web en indiquant d'abord les suites cryptographiques les plus robustes, classées par ordre de préférence décroissant en fonction du niveau de sécurité qu'elles offrent.

Il est donc recommandé de configurer le serveur Web en indiquant d'abord les suites cryptographiques les plus robustes, classées par ordre de préférence décroissant en fonction du niveau de sécurité qu'elles offrent

3.5.5 Robustesse des algorithmes et des suites cryptographiques

Les suites cryptographiques utilisées par TLS définissent la manière dont toutes les communications entre le client et le serveur seront effectuées via HTTPS (à l'aide de plusieurs composants ou algorithmes cryptographiques), y compris le processus d'authentification pour confirmer que la communication est faite avec le serveur ou l'application Web légitime, jusqu'à l'échange de données chiffrées ultérieur.

Si l'un des algorithmes cryptographiques utilisés est faible ou peu sûr, il doit être remplacé par un algorithme plus moderne. Les algorithmes cryptographiques suivants ne devraient pas être utilisés de nos jours :

- Le chiffreur[72] (et le mécanisme d'authentification) NULL, car il ne fournit pas de capacités de chiffrement (et d'authentification).
- Les suites basées sur le protocole Diffie-Hellman anonyme (ADH, Anonymous Diffie-Hellman), car elles ne fournissent pas de capacités d'authentification..
- Les suites cryptographiques avec des algorithmes de chiffrement faibles basés sur des clés dont la forteresse cryptologique est inférieure à 100 bits.
- Les suites cryptographiques dont les algorithmes de chiffrement sont autorisés à être exportés (voir l'attaque FREAK dans la section "3.8. Vulnérabilités dans HTTPS").
- L'algorithme de chiffrement RC4, car il n'est pas sûr.
- L'algorithme de chiffrement DES, car il est peu sûr et lent, et sa variante Triple DES (TDES ou 3DES).
- Les algorithmes de *hachage* MD5, RIPMED-160 et SHA-1, car ils sont peu sûrs ou faibles.

.

TLS peut être considéré comme un cadre qui permet de négocier et d'utiliser différents protocoles cryptographiques, en sélectionnant les composants et les paramètres qui définiront le type de sécurité à utiliser. Les suites cryptographiques utilisées par TLS définissent dans leur nom un ensemble d'algorithmes cryptographiques associés aux différents composants et phases du protocole TLS. Parmi eux, la méthode d'échange de clés, la méthode d'authentification (clés et certificats), l'algorithme de chiffrement symétrique et la longueur des clés de chiffrement à utiliser sont spécifiés dans l'ordre suivant, comme le montre la figure suivante [Réf. - 10]), ainsi que leur mode d'utilisation (dans le cas où il en existe plusieurs) et l'algorithme de *hachage* pour le MAC (ou l'algorithme de fonction pseudo-aléatoire, PRF, utilisé par exemple dans les algorithmes de type AEAD, qui ne font pas appel à un MAC au niveau TLS) :

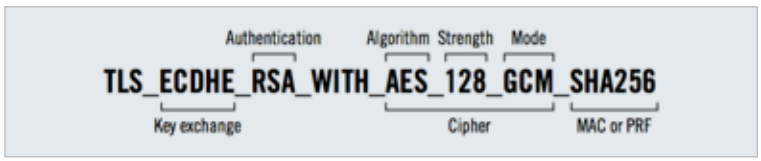


Figure 5.
Nomenclature standard (IANA) des suites cryptographiques utilisées par TLS.
Source : [Réf. - 10].

Par exemple, les suites cryptographiques suivantes utilisent les algorithmes détaillés ci-dessous. Toutes les suites cryptographiques du protocole TLS commencent par "TLS" :

[72] Le terme "chiffreur" est parfois utilisé pour abréger le terme "algorithme de chiffrement".

Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS avec échange de clés ECDHE, authentification par clés (et certificats numériques) ECDSA, algorithme de chiffrement AES 128 bits en GCM (Galois/Counter Mode) et SHA-256 comme fonction pseudo-aléatoire (PRF).

Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS avec échange de clés DHE, authentification par clés (et certificats numériques) RSA, algorithme de chiffrement AES 256 bits en mode CBC (*Cipher Block Chaining*) et SHA-1 comme algorithme de *hachage* pour le calcul du MAC.

La partie la plus complexe est celle qui est associée à la partie finale du nom, c'est-à-dire l'algorithme de *hachage* pour le calcul du MAC, ou l'algorithme de la fonction pseudo-aléatoire (PRF), car dans TLS 1.2 et en raison de la flexibilité qu'offre cette version du protocole, il peut avoir différentes significations. Par exemple, pour les suites AEAD (telles que AES/GCM et ChaCha20-Poly1305), la dernière partie représente la fonction PRF. Pour les suites définies avant TLS 1.2, le nom de la suite définit l'algorithme de *hachage* pour le (calcul du) MAC dans TLS (p. ex. SHA-1), la fonction PRF n'est pas référencée, car c'est la version du protocole (non référencée dans le nom de la suite) qui la définit, puisque, par exemple, une suite peut utiliser HMAC-SHA256 comme PRF pour TLS 1.2, mais HMAC-SHA1 comme PRF pour TLS 1.0 [Réf. - 10].

Il faut noter qu'OpenSSL utilise une nomenclature légèrement différente pour référencer les suites cryptographiques que celle définie dans la norme IANA [Réf. - 23] et utilisée par d'autres technologies Web, comme IIS. Vous trouverez ci-dessous deux exemples de chaque format de nomenclature (qui, comme vous pouvez le constater, sont assez similaires et font référence au même code ou identifiant unique de chaque suite cryptographique) :

Code: **0xC0,0x2B ó 0xC02B (RFC 5289)**
OpenSSL: **ECDHE-ECDSA-AES128-GCM-SHA256**
IANA: **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256**

Code: **0x00,0x9E ó 0x009E (RFC 5288)**
OpenSSL: **DHE-RSA-AES128-GCM-SHA256**
IANA: **TLS_DHE_RSA_WITH_AES_128_GCM_SHA256**

REMARQUE :
Une comparaison de la nomenclature IANA avec celle utilisée par d'autres bibliothèques SSL/TLS, telles que GnuTLS, NSS et OpenSSL, est disponible.[71]

Il est donc recommandé d'utiliser des suites cryptographiques commençant par "ECDHE-RSA-..." (de préférence) ou "DHE-RSA-...", en cas d'utilisation de clés cryptographiques RSA, et "ECDHE-ECDSA-...", en cas d'utilisation de clés cryptographiques ECDSA (voir section "3.2. Génération des clés cryptographiques"). Le document "ANNEXE B. SUITES CRYPTOGRAPHIQUES RECOMMANDÉES" fournit une liste de référence des suites cryptographiques préférées pour une configuration initiale de serveur Web HTTPS (au format OpenSSL).

Il convient de noter que (dans l'avenir), lors de l'utilisation du protocole TLS 1.3, l'algorithme d'échange de clés n'est pas négocié dans le *cadre* des suites cryptographiques, mais séparément (seuls les algorithmes qui font usage du *Forward Secrecy*, tels que ECDHE et FFDHE, *Finite Field Diffie-Hellman Ephemeral*, RFC 7919 d'août 2016, sont pris en charge dans TLS 1.3). Par conséquent, les suites cryptographiques prises en charge par TLS 1.3[73] ne contiennent pas l'algorithme d'échange de clés à utiliser dans leur nom ou dans leur définition. Par exemple :

Il convient de noter que (dans l'avenir), lors de l'utilisation du protocole TLS 1.3, l'algorithme d'échange de clés n'est pas négocié dans le cadre des suites cryptographiques, mais séparément

Code:	**0x13,0x01 ó 0x1301 (RFC5116)**
IANA:	**TLS_AES_128_GCM_SHA256**
Code:	**0x13,0x02 ó 0x1302 (RFC5116)**
IANA:	**TLS_AES_256_GCM_SHA384**
Code:	**0x13,0x03 ó 0x1303 (RFC7539)**
IANA:	**TLS_CHACHA20_POLY1305_SHA256**

TLS peut utiliser de nombreux algorithmes de chiffrement, à la fois par flux (*stream*), par bloc (*block*) et par authentification (AEAD) dans TLS 1.2, tels que 3DES, RC4, AES, CAMELLIA, ChaCha20, etc. Parmi les algorithmes de chiffrement, il est recommandé d'utiliser ceux qui fournissent un chiffrement authentifié (AE), c'est-à-dire qui combinent nativement le chiffrement et l'intégrité avec l'authentification des données, communément appelé AEAD (*Authenticated Encryption with Associated Data*). Ces algorithmes modernes utilisent également des clés fortes et le *Forward Secrecy* (voir section "3.5.1. *Forward Secrecy*"). Par exemple, l'AES en mode GCM (Galois/Counter Mode), par opposition aux modes CBC (*Cipher Block Chaining*) moins sûrs, offre ces caractéristiques.

[73] https://tlswg.github.io/tls13-spec/#rfc.appendix.B.4

Du point de vue des performances, de nombreux CPU supportent aujourd'hui des opérations qui permettent d'exécuter l'algorithme AES sur matériel ou hardware (AES-NI[74]), ce qui accélère et optimise les opérations de chiffrement et déchiffrement.

La validation de l'intégrité des données échangées dans TLS, par le biais d'un MAC (*Message Authentication Code*), est implicite dans le processus de chiffrement.

La validation de l'intégrité des données échangées dans TLS, par le biais d'un MAC (*Message Authentication Code*), est implicite dans le processus de chiffrement

Après avoir obtenu la clé de chiffrement maître, celle-ci (ainsi que deux graines aléatoires du client et du serveur) est traitée par une fonction pseudo-aléatoire (PRF) pour obtenir le matériel cryptographique nécessaire en fonction de la suite cryptographique à utiliser, et principalement des algorithmes de chiffrement. Comme chaque algorithme de chiffrement nécessite d'un matériel cryptographique de longueur différente, la valeur de sortie sera adaptée en fonction de la suite négociée.

Il est donc recommandé d'utiliser le chiffrement AEAD en utilisant le chiffrement AES 128 bits ou 256 bits GCM pour TLS (1.2), tel que défini dans la norme RFC 5288[75], daté d'août 2008. En outre, et une fois que son utilisation aura été normalisée dans TLS, le chiffreur de flux ChaCha20 pourra être utilisé, ainsi que l'authentificateur Poly1305, tel que défini dans la norme RFC 7905[76], daté de juin 2016.

Du point de vue des navigateurs et clients Web, il est possible d'évaluer leur prise en charge d'algorithmes et de suites cryptographiques robustes ou faibles (vulnérables) grâce au service "SSL Client Test" de Qualys SSL Labs [Réf. - 3], et du point de vue d'une mauvaise implémentation, grâce au service "BadSSL" [Réf. - 20] et, en particulier, à son "Dashboard"[77]. Le service "BadSSL" permet de vérifier le comportement des clients Web face à des configurations HTTPS incorrectes ou non souhaitées. Le domaine associé, "badssl.com", fournit une multitude de serveurs (et de noms d'hôtes) différents, chacun d'entre eux présentant des faiblesses et des vulnérabilités spécifiques, permettant de vérifier les capacités de connexion HTTPS d'un client Web dans ce type de scénarios vulnérables, ainsi que d'obtenir (du point de vue de l'utilisateur) le message d'erreur, l'alerte ou l'avertissement spécifique généré dans chaque situation indésirable.

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

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

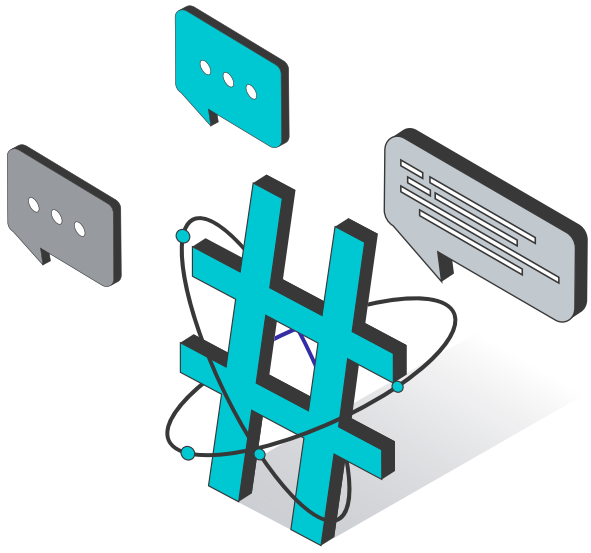
76 "ChaCha20-Poly1305 Cipher Suites for TLS": https://tools.ietf.org/html/rfc7905

77 https://badssl.com/dashboard/

En résumé, une fois de plus, il faut toujours rechercher un équilibre entre sécurité et performances. Ainsi, à l'heure actuelle, pour les environnements Web standard, il est recommandé d'utiliser des clés ECDSA de 256 bits et des clés RSA de 2.048 bits pour la génération de clés cryptographiques (voir section "3.2. Génération des clés cryptographiques"), pour l'échange de clés, il est recommandé d'utiliser des clés ECDHE de 256 bits et DHE de 2.048 bits (voir section "3.5.2. Échange de clés robuste") et, à titre de recommandation générale, il convient d'utiliser des algorithmes de chiffrement symétrique offrant une sécurité d'au moins 128 bits (l'impact sur les performances des algorithmes de chiffrement symétrique de 256 bits doit être évalué par rapport à l'augmentation de la sécurité qu'ils offrent). En outre, à titre de recommandation générale, il convient d'utiliser des algorithmes de hachage basés sur SHA-256 et des algorithmes plus robustes (p. ex. SHA-384, SHA-512...) dans des cas très spécifiques qui exigent un degré de sécurité plus élevé.

Il faut noter que les suites cryptographiques qui utilisent l'algorithme de hachage SHA-1 pour le MAC (référencé à la fin de son nom par "SHA"), l'utilisent en fait avec une clé dans sa variante HMAC (*keyed-Hash Message Authentication Code*), c'est-à-dire HMAC-SHA1, pour laquelle il n'y a pas d'attaques connues et qui peut donc être utilisée avec des garanties de sécurité (contrairement à l'utilisation de SHA-1 sans HMAC).

À titre de recommandation générale, il convient d'utiliser des algorithmes de hachage basés sur SHA-256 et des algorithmes plus robustes dans des cas très spécifiques qui exigent un degré de sécurité plus élevé



3.5.6 Recommandations relatives aux algorithmes et aux suites cryptographiques

Voici le résumé des recommandations relatives aux algorithmes et aux suites cryptographiques :

- Utilisez des algorithmes d'échange de clés qui garantissent le *Forward Secrecy*, de préférence ECDHE, ou alternativement DHE, par opposition à RSA.
- Il convient d'utiliser des algorithmes d'échange de clés de 256 bits dans le cas de l'ECDHE et de 2.048 bits dans le cas du DHE.
- Si ECDHE est utilisé, le type de courbes elliptiques à utiliser doit être évalué en détail. Il existe des courbes largement supportées, comme secp256r1(P-256) et secp384r1 (P-384), définies par le NIST et qui n'ont pas fait l'objet d'une analyse publique exhaustive, et de nouvelles courbes considérées comme robustes, comme Curve25519 (X2559) et Curve448 (X448), dont le support commence à devenir populaire.
- Dans le cas de l'utilisation de DHE, la taille des paramètres DH doit être d'au moins 2.048 bits, et il est recommandé de la faire coïncider avec la longueur ou à la forteresse de la clé privée correspondante.
- Dans le cas de l'utilisation de DHE, il n'est pas recommandé d'utiliser des groupes DH connus ou préconfigurés, tels que ceux de la norme RFC 3526, ou de générer vos propres groupes. Il est préférable d'utiliser les groupes DH prédéfinis dans la norme RFC 7919 : ffdhe2048, ffdhe3072 ou ffdhe4096.
- Configurez le serveur Web en donnant la préférence aux suites cryptographiques les plus robustes, classées par ordre décroissant de préférence en fonction du niveau de sécurité qu'elles offrent.
- Ne pas utiliser le chiffreur NULL, les suites basées sur ADH, les algorithmes de chiffrement avec des clés dont la robustesse cryptologique est inférieure à 100 bits, les suites avec des algorithmes de chiffrement autorisés à l'exportation, les chiffreurs RC4, DES, 3DES (ou TDES), les algorithmes de hachage MD5, RIPMED-160 et SHA-1, etc.

- Lors de la hiérarchisation des suites cryptographiques TLS, en ce qui concerne l'échange de clés et l'algorithme d'authentification, il est recommandé d'utiliser les suites cryptographiques commençant par "ECDHE-ECDSA-...", en cas d'utilisation de clés cryptographiques ECDSA, et par "ECDHE-RSA-..." (de préférence) ou "DHE-RSA-...", en cas d'utilisation de clés cryptographiques RSA.

- Utilisez les algorithmes de chiffrement qui fournissent un chiffrement authentifié (AE) dans TLS 1.2, communément appelé AEAD (*Authenticated Encryption with Associated Data*), tels que les modes GCM (Galois/Counter Mode) de l'AES 128 bits ou 256 bits, par opposition aux modes CBC (*Cipher Block Chaining*) de l'AES, ou le chiffreur de flux ChaCha20, ainsi que l'authentificateur Poly1305.

- Il convient d'utiliser des algorithmes de chiffrement symétrique offrant une sécurité d'au moins 128 bits (l'impact sur les performances des algorithmes de chiffrement symétrique de 256 bits doit être évalué par rapport à l'augmentation de la sécurité qu'ils offrent).

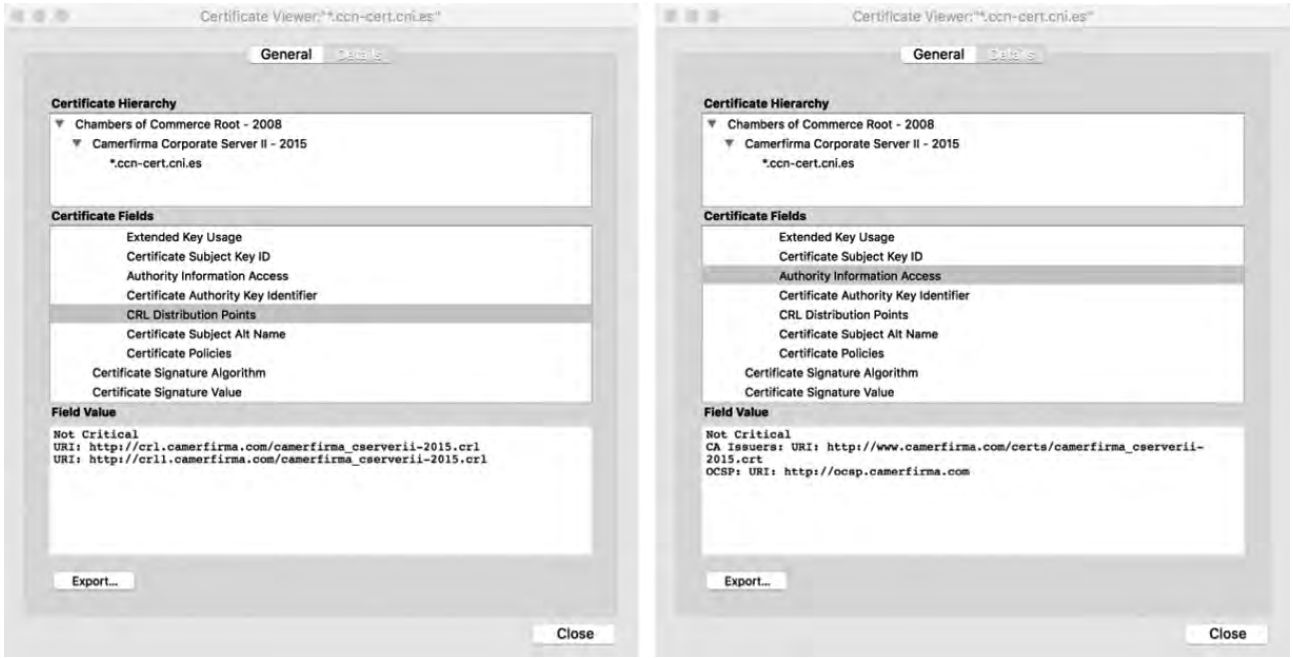
- Les algorithmes de hachage basés sur SHA-256 doivent être utilisés, et des algorithmes plus robustes (p. ex. SHA-384, SHA-512...) doivent être utilisés dans des cas très spécifiques, qui nécessitent un degré de sécurité plus élevé.

3.6. Mécanismes de révocation des certificats numériques

Les mécanismes de révocation des certificats numériques permettent au propriétaire du certificat d'indiquer aux utilisateurs (navigateurs et clients Web) que le certificat n'est plus valide, car il a probablement été remplacé par un nouveau.

Les certificats numériques doivent inclure des mécanismes de révocation tels que les CRL (listes de révocation de certificats) ou l'OCSP (*Online Certificate Status Protocol*) ou, de préférence, l'*OCSP Stapling*. Le certificat numérique lui-même fournit des instructions ou des références pour utiliser les deux premiers mécanismes afin de vérifier la validité du certificat et s'il a été révoqué ou non :

Figure 6.
Informations de révocation comprises dans les certificats numériques.



3.6.1 *Certificate Revocation Lists* (CRLs)

Le mécanisme CRL (*Certificate Revocation List* ou Liste de Révocation de Certificats), RFC 6818[78], permet aux navigateurs et clients Web de vérifier la validité de l'état d'un certificat numérique en téléchargeant la liste de révocation (ci-après liste CRL) qui est maintenue par l'AC qui a émis le certificat. Si le numéro de série du certificat figure dans la liste, cela signifie qu'il a été révoqué.

La référence à la liste CRL se trouve dans le champ "CRL Distribution Points" du certificat numérique.

Les principales limites de la CRL sont son évolutivité, en raison de la taille des listes de révocation, qui augmentent avec chaque nouveau certificat révoqué, l'absence d'informations récentes ou mises à jour (étant donné qu'en raison de la taille des listes, les clients Web conservent généralement une copie locale en cache), les implications en termes de performances lors de la réalisation du *handshake* TLS et de la vérification du certificat (étant donné qu'il est nécessaire de télécharger la liste CRL au moment de la connexion), et la gestion des échecs de téléchargement de la liste CRL.

Si, pour une raison quelconque, il n'est pas possible de télécharger la liste CRL, il est courant que les clients Web utilisent le mode *soft fail* et considèrent le certificat comme valide.

L'utilisation des CRL par les navigateurs et clients Web tend aujourd'hui à la désuétude, ou à des scénarios d'utilisation très spécifiques (p. ex. les certificats EV) ou peu répandus, presque toujours utilisés en dernier recours pour la vérification.

Le mécanisme CRL permet aux navigateurs et clients Web de vérifier la validité de l'état d'un certificat numérique en téléchargeant la liste de révocation qui est maintenue par l'AC qui a émis le certificat

[78] https://tools.ietf.org/html/rfc6818

3.6.2 *Online Certificate Status Protocol* (OCSP)

Le mécanisme OCSP (*Online Certificate Status Protocol* ou Protocole de Vérification de Certificat En Ligne), RFC 6960[79], permet aux navigateurs et clients Web de vérifier la validité de l'état d'un certificat numérique en effectuant une requête en temps réel auprès du service OCSP (appelé service de réponse OCSP ou *OCSP Responder*) géré par l'AC qui a émis le certificat. La requête au service OCSP est effectuée uniquement pour le numéro de série du certificat, ce qui optimise la bande passante nécessaire à la vérification de l'état de révocation, et la réponse obtenue permet de savoir s'il a été révoqué ou non.

Le mécanisme OCSP permet aux navigateurs et clients Web de vérifier la validité de l'état d'un certificat numérique en effectuant une requête en temps réel auprès du service OCSP géré par l'AC qui a émis le certificat

La référence au service OCSP se trouve dans le champ "(Certificate) Authority Information Access" (dans la ligne "OCSP") du certificat numérique.

Les principales limites de l'OCSP sont les suivantes : l'AC doit disposer d'un service OCSP rapide, fiable et à jour, toujours disponible et capable de gérer des pics de travail élevés, et encore, les implications en termes de performances lors de l'achèvement de du *handshake* TLS et de la vérification du certificat (étant donné qu'il est nécessaire de recevoir la réponse du service OCSP au moment de la connexion), ainsi que la gestion des défaillances d'accès au service OCSP. D'autre part, et du point de vue du respect de la vie privée, l'autorité de certification peut savoir quels sites Web sont visités par un client Web.

Si, pour une raison quelconque, il n'est pas possible d'accéder au service OCSP, il est courant que les clients Web utilisent le mode *soft fail* et considèrent le certificat comme valide. Il est relativement facile de mener des attaques (via des erreurs TCP, des erreurs HTTP ou même des erreurs dans le propre protocole OCSP) pour empêcher le client Web de recevoir la réponse du service OCSP.

Les réponses OCSP peuvent être valables jusqu'à 10 jours pour les certificats numériques finaux (généralement 7 jours), et jusqu'à 12 mois pour les certificats numériques des AC intermédiaires. En outre, certaines optimisations des performances appliquées à OCSP facilitent les attaques par *replay*.

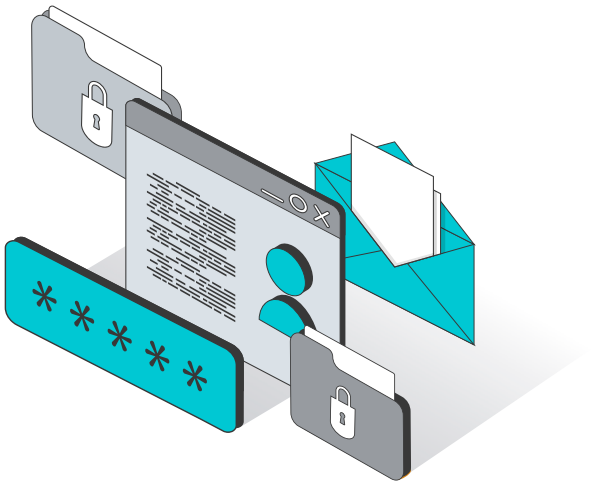
[79] https://tools.ietf.org/html/rfc6960

Au moment de choisir une AC, il est recommandé de vérifier que l'AC utilisée pour la délivrance des certificats numériques (voir section "3.3.2. Délivrance des certificats numériques : Les Autorités de certification (AC)") dispose d'un service OCSP rapide et fiable.

Là encore, comme pour les CRL, l'utilisation d'OCSP par les navigateurs et clients Web est aujourd'hui très variée, et il n'est généralement pas actif par défaut, ou seulement dans des scénarios d'utilisation très spécifiques (p. ex. les certificats EV) ou peu répandus, de sorte qu'il ne peut être considéré comme efficace, comme le montre l'apparition d'autres alternatives propriétaires telles que CRLSets dans Chrome ou OneCRL dans Firefox.

Si un client Web utilise OCSP, chaque fois qu'il se connecte au serveur Web, il devra également se connecter à l'AC pour vérifier la validité actuelle du certificat numérique. L'*OCSP Stapling* (voir section "3.6.3. *OCSP Stapling*") permet d'éviter cette dépendance continue à l'AC (avec les implications positives associées en termes de disponibilité, de performances et de confidentialité).

Si un client Web utilise OCSP, chaque fois qu'il se connecte au serveur Web, il devra également se connecter à l'AC pour vérifier la validité actuelle du certificat numérique



3.6.3 *OCSP Stapling* et *OCSP Must-Staple*

Si un client Web prend en charge et demande l'*OCSP Stapling*, et que le serveur Web met également en œuvre l'*OCSP Stapling*, chaque fois que le client se connecte au serveur Web, ce dernier peut fournir directement la réponse OCSP afin que le client puisse vérifier la validité actuelle du certificat numérique, évitant ainsi la plupart des conséquences négatives associées à l'utilisation d'OCSP ou de CRL.

L'*OCSP Stapling*[80], RFC 6066[81], a été conçu pour combler les lacunes de CRL et OCSP, qui ne fournissaient pas la sécurité et les performances souhaitées au moment de vérifier la révocation des certificats sur les connexions HTTPS. Dans la plupart des cas, les contrôles de révocation n'ont pas été efficaces, comme cela a été démontré en 2011 lorsqu'il a été nécessaire de distribuer des mises à jour aux navigateurs et clients Web afin d'ajouter aux listes noires de révocation les faux certificats générés après la compromission de plusieurs AC. Avec l'*OCSP Stapling*, c'est le serveur Web (et non pas le client) qui vérifie périodiquement l'état de révocation et fournit les détails au client.

L'extension TLS associée à l'*OCSP Stapling* permet au serveur Web de fournir au client Web les informations les plus récentes possibles sur l'état de révocation d'un certificat numérique, envoyées (attachées ou agrafées, d'où son nom) lors de l'établissement de la connexion ou du *handshake* TLS, depuis le serveur Web lui-même (sans dépendre de l'AC au moment de la connexion). Pour ce faire, le serveur obtient périodiquement l'état du certificat au moyen d'une requête OCSP adressée au service de l'AC, et reçoit une réponse signée par l'AC et comportant un horodatage (*timestamp*). Lorsque le client Web reçoit cette réponse pendant l'établissement de la connexion TLS, il peut vérifier la signature de la réponse OCSP et donc la validité du certificat. Le client peut également faire confiance à la réponse, car elle a une validité limitée dans le temps (basée sur l'horodatage).

Si pour une raison quelconque, il n'est pas possible d'obtenir la réponse *OCSP Stapling*, et que le serveur Web déclare utiliser ce mécanisme de révocation (via la directive *OCSP Must-Staple*, décrite ci-dessous), les clients Web peuvent utiliser le mode *hard fail* et ne pas considérer le certificat comme valide.

Si un client Web prend en charge et demande l'*OCSP Stapling*, et que le serveur Web met également en œuvre l'*OCSP Stapling*, chaque fois que le client se connecte au serveur Web, ce dernier peut fournir directement la réponse OCSP afin que le client puisse vérifier la validité actuelle du certificat numérique

[80] https://scotthelme.co.uk/ocsp-stapling-speeding-up-ssl/

[81] https://tools.ietf.org/html/rfc6066

Dans le scénario initial d'utilisation de l'*OCSP Stapling*, le client Web ne peut pas savoir si le site Web prend en charge l'*OCSP Stapling* et doit donc s'attendre à recevoir une réponse OCSP lors du *handshake* TLS. Pour résoudre ce scénario, l'*OCSP Must-Staple* a été créée.

L'*OCSP Must-Staple*[82], RFC 7633[83], définit un nouveau paramètre associé aux certificats numériques X.509 qui doivent être activés par l'AC, indiquant au navigateur ou au client Web que le certificat numérique doit être proposé dans le *cadre* d'une session TLS incluant une réponse d'*OCSP Stapling*. Sinon, le client Web doit considérer que le certificat numérique n'est pas valide (*hard fail*).

Le processus utilisé pour générer une demande de signature de certificat (CSR) à l'AC doit inclure l'extension *OCSP Must-Staple*.

La combinaison de l'*OCSP Stapling* et de l'*OCSP Must-Staple* étant plus restrictive que les options de révocation disponibles auparavant, et l'apparition de scénarios potentiels pouvant affecter la disponibilité de l'environnement Web s'ils ne sont pas mis en œuvre correctement (comme avec HPKP ou CSP), il est recommandé de l'implémenter avec prudence et d'évaluer sa mise en œuvre en détail. À cette fin, un nouveau mécanisme de notification, appelé *OCSP Expect-Staple*[84], est disponible et permet au propriétaire d'un site Web de surveiller le fonctionnement de l'*OCSP Stapling* avant son activation stricte via *OCSP Must-Staple*. Ses capacités de surveillance sont similaires à celles associées à la directive "report-uri" des en-têtes de sécurité HTTP tels que HPKP (voir section "3.3.9 Restriction des certificats numériques légitimes : HPKP") ou CSP (voir section "3.7.6. *Content Security Policy* (CSP)").

À cette fin, un nouveau mécanisme de notification, appelé *OCSP Expect-Staple*, est disponible et permet au propriétaire d'un site Web de surveiller le fonctionnement de l'*OCSP Stapling* avant son activation stricte via *OCSP Must-Staple*

La liste préchargée des HSTS (voir section "3.7.4. Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS") devient l'emplacement de référence dans les navigateurs Web pour représenter les exigences de sécurité des environnements Web. Cette liste contient non seulement la directive HSTS elle-même, mais aussi les épingles préfixées de HPKP (pour certains domaines seulement), *Expect-CT* (voir "3.3.6. *Certificate Transparency* (CT)") et *OCSP Expect-Staple*. Au moment de la rédaction de ce document, le processus d'ajout de l'*OCSP Expect-Staple* à l'environnement Web se fait manuellement avec les responsables de la liste préchargée de Chromium/Chrome.

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

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

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

La définition de l'*OCSP Expect-Staple* dans cette liste comprend : la déclaration selon laquelle les réponses de l'environnement Web doivent utiliser l'*OCSP Stapling* et inclure une réponse OCSP, l'URL à laquelle soumettre les rapports d'erreur associés à *OCSP Expect-Staple*, et si la politique s'applique également à tous les sous-domaines (includeSubDomains).

Lorsque la politique *OCSP Expect-Staple* est active, si le navigateur Web établit une connexion avec l'environnement Web mais ne reçoit pas de réponse OCSP valide via l'*OCSP Stapling*, il génère un rapport notifiant cette situation à l'URL préalablement définie, en utilisant le format décrit dans la spécification[85]. Le rapport notifie non seulement si la réponse OCSP n'a pas été reçue (absente), mais aussi si elle a été reçue mais est invalide pour d'autres raisons (expirée, non applicable au certificat numérique reçu, format incorrect, etc.) Ces détails fournissent les informations nécessaires pour configurer et mettre en œuvre un environnement d'OCSP optimal.

Initialement, l'OCSP ne pouvait fournir que des informations sur le certificat numérique du serveur Web, mais la norme RFC 6961[86] (de juin 2013) étend ses capacités pour fournir les détails de révocation de plusieurs certificats simultanément, tels que ceux associés aux AC intermédiaires représentées dans la chaîne de certification.

L'avantage d'utiliser l'OCSP conjointement avec l'*OCSP Must-Staple* est que le délai dans lequel les clients Web sont informés d'une compromission potentielle d'un certificat numérique est considérablement réduit, passant d'un maximum de 39 mois (durée de validité maximale des certificats numériques, ou 825 jours à partir de mars 2018, selon la nouvelle proposition, voir section "3.3.5. Renouvellement des certificats numériques"), au maximum associé à la durée de vie d'une réponse OCSP, c'est-à-dire normalement un maximum de 7 jours.

L'avantage d'utiliser l'OCSP conjointement avec l'*OCSP Must-Staple* est que le délai dans lequel les clients Web sont informés d'une compromission potentielle d'un certificat numérique est considérablement réduit

[85] https://docs.google.com/document/d/1aISglJIIwglcOAhqNfK-2vtQl-_dWAapc-VLDh-9-BE/edit#heading=h.rkpittae54q

[86] https://tools.ietf.org/html/rfc6961

3.6.4 Recommandations pour la révocation des certificats numériques

Le résumé des recommandations pour la révocation des certificats numériques comprend :

- Les certificats numériques doivent inclure des mécanismes de révocation tels que les CRL (listes de révocation de certificats) ou l'OCSP (*Online Certificate Status Protocol*) ou, de préférence, l'*OCSP Stapling*.
- Au moment de choisir une AC, il est recommandé de vérifier qu'elle dispose d'un service OCSP rapide et fiable.
- Utilisez l'extension *TLS OCSP Stapling* pour fournir aux clients des informations à jour sur l'état de révocation d'un certificat numérique.
- Utilisez une AC qui permet d'ajouter la politique *OCSP Must-Staple* aux certificats numériques, pour permettre aux clients Web de considérer le certificat comme invalide s'il n'est pas possible d'obtenir une réponse OCSP agrafée valide.
- Utiliser l'*OCSP Must-Staple*. En complément, utilisez les capacités de notification *OCSP Expect-Staple*, qui permettent de contrôler le fonctionnement de l'OCSP avant son activation stricte par *OCSP Must-Staple*. Ces capacités, encore disponibles à titre expérimental, permettent une utilisation progressive et contrôlée de l'*OCSP Must-Staple*.
- Fournir les détails de révocation de plusieurs certificats simultanément, tels que ceux associés aux AC intermédiaires, par OCSP (si cela est jugé nécessaire).

3.7. Contenu et applications Web HTTPS

Les sections suivantes fournissent de nombreuses recommandations qui affectent la conception et la configuration du serveur Web et de son contenu, ainsi que des applications Web. Ces sections sont axées sur une couverture maximale pour offrir tous les contenus disponibles dans l'environnement Web via HTTPS.

3.7.1 Priorité dans les navigateurs Web

L'un des objectifs de la plupart des sites Web est d'être bien positionné dans les navigateurs Internet (tels que Google, Yahoo !, Bing, etc.) afin d'être facilement trouvé et identifié par les utilisateurs potentiels.

En août 2014, Google a annoncé [Réf. - 24] qu'il accorderait plus de pertinence dans le positionnement (ou le classement) des recherches, associé aux algorithmes de SEO (*Search Engine Optimisation* et aux sites Web qui font usage de HTTPS par rapport à ceux qui utilisent HTTP. Par la suite, en décembre 2015, Google a annoncé que la recherche et l'indexation des pages Web seraient effectuées par défaut pour la version HTTPS, c'est-à-dire en commençant par les pages HTTPS par rapport aux pages HTTP[87], même si aucune référence au HTTPS n'a été identifiée.

Pour cette raison, afin de faciliter l'indexation des contenus du site Web, il est recommandé de ne pas ajouter la directive "noindex" pour les URL qui utilisent le HTTPS, ni à travers la balise HTML "<meta>", ni à travers les en-têtes HTTP.

L'un des objectifs de la plupart des sites Web est d'être bien positionné dans les navigateurs Internet afin d'être facilement trouvé et identifié par les utilisateurs potentiels

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

En outre, la politique établie dans le fichier "robots.txt" doit permettre l'indexation du contenu HTTPS du serveur Web par les navigateurs Web.

Également lié aux navigateurs Web et aux algorithmes de Google (SEO), en ce qui concerne l'existence de contenu dupliqué et la redirection de toute ressource de HTTP à HTTPS (voir section "3.7.3. Forcer l'utilisation (par défaut) de HTTPS"), il est recommandé de faire usage de la balise "rel=canonical"[88] [89], qui permet de définir quelle est la ressource canonique ou de référence au sein du serveur Web pour des contenus similaires (ou dupliqués).

La balise "rel=canonical" doit être ajoutée à l'en-tête "<head>" de toutes les pages Web dupliquées (et même des pages Web uniques ou non dupliquées), en spécifiant la page Web HTTPS de référence pour les navigateurs Web :

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

Dans le cas où la redirection globale de HTTP vers HTTPS ne s'applique pas à toutes les ressources de l'application Web, cette balise permet d'ajouter une indication supplémentaire faisant référence à l'intention d'utiliser HTTPS, pour l'indexation par les navigateurs Web.

Figure 7.
Priorité des sites Web HTTPS dans les navigateurs Web.
Source : Google

[88] https://webmasters.googleblog.com/2009/02/specify-your-canonical.html

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

3.7.2 Performances

Dans le passé, l'un des arguments contre l'utilisation de HTTPS au lieu de HTTP était l'impact sur les performances des serveurs ou des applications Web. De nombreuses améliorations et optimisations dans les protocoles associés à HTTPS, ainsi que dans les implémentations [Réf. - 9], ont augmenté de manière significative les performances de HTTPS et le temps de chargement des pages Web qui utilisent ce protocole, et lui ont même permis de dépasser les performances de HTTP, surtout si HTTP/2 est utilisé (voir section "3.4.5. HTTP/2").

Parmi les améliorations et optimisations des performances (appliquées à l'utilisation du CPU, à la latence de la connexion, etc.) qui se trouvent disponibles, nous citons : le *TLS handshake* (1-RTT), le *TLS record size* et la chaîne de certification, le *TLS session resumption* ou le *TLS session caching* (*TLS session tickets*), le *TLS False Start*, le *TCP Fast Open*, le *TLS early termination*, l'OCSP, etc., ainsi que les nouvelles fonctionnalités qui seront intégrées dans TLS 1.3, telles que le 0-RTT ou le zéro-RTT.

De nombreuses ressources sont disponibles[90] pour évaluer les performances de HTTP par rapport à HTTPS, comme le "Test HTTP vs HTTPS"[91].

En raison d'attaques telles que CRIME (voir section "3.8. Vulnérabilités dans HTTPS"), il est recommandé de désactiver les capacités de compression TLS du serveur Web. Cependant, il existe la possibilité d'utiliser les mécanismes de compression Brotli [Réf. - 25], mais seulement pour les connexions HTTPS, où l'en-tête HTTP "Accept-encoding : br" permet au client Web d'indiquer qu'il est capable de les supporter. Ces fonctionnalités étaient incluses dans Firefox 44[92], Chrome 55[93], Edge 15 ou Opera 38[94]. Brotli apporte des améliorations d'environ 20 à 26 % par rapport aux autres algorithmes de compression existants, ce qui permet de réduire la bande passante utilisée pour les communications Web et d'accélérer le temps de chargement des pages et des ressources Web. Il est recommandé de vérifier si le support de la compression Brotli est disponible sur le serveur Web, et de l'activer si c'est le cas.

De nombreuses améliorations et optimisations dans les protocoles associés à HTTPS, ainsi que dans les implémentations [Réf. - 9], ont augmenté de manière significative les performances de HTTPS et le temps de chargement des pages Web qui utilisent ce protocole

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

91 https://www.httpvshttps.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

3.7.2.1 *TLS FALSE START*

L'établissement d'une session TLS, via un *TLS handshake* standard, nécessite l'utilisation de 2-RTT (*roundtrips* ou échanges aller-retour) entre le client et le serveur [Réf. - 9], ce qui affecte les performances des connexions HTTPS.

TLS False Start est une extension du protocole TLS qui permet la transmission de données d'application chiffrées lorsque la session TLS n'est que partiellement établie, réduisant ainsi l'établissement de la session TLS à 1-RTT. *TLS False Start* optimise les performances TLS en sacrifiant légèrement la sécurité TLS, car les données sont envoyées avant que l'intégrité du *TLS handshake* ne soit vérifiée.

TLS False Start est pris en charge par tous les navigateurs et clients Web modernes, mais pour être utilisés en toute confiance et pour compenser le sacrifice de sécurité mentionné ci-dessus, ils vérifient que le serveur Web utilise (au moins) des suites de chiffrement de 128 bits, avec Forward Secrecy, et l'extension TLS NPN[95] (avant *Next Protocol Negotiation*) ou ALPN (*Application Layer Protocol Negotiation*).

Si vous combinez le *TLS False Start* avec le *TLS session resumption* (ou *TLS session caching*, voir section "3.7.2.2. *TLS session resumption* ou *TLS session caching*, et *TLS session tickets*" il est possible d'utiliser les *TLS handshake* 1-RTT pour les nouvelles sessions et d'optimiser les calculs cryptographiques pour les sessions suivantes (résumées).

En outre, et afin de pouvoir comparer les versions du protocole TLS entre elles, l'un des objectifs de conception de TLS 1.3 (voir section "3.4.1. Versions des protocoles SSL et TLS") est de disposer des capacités permettant d'établir de nouvelles sessions TLS en 1-RTT et de résumer les sessions en 0-RTT.

3.7.2.2 *TLS SESSION RESUMPTION* OU *TLS SESSION CACHING,* ET *TLS SESSION TICKETS*

Si un client a déjà communiqué avec le serveur via TLS, il est possible d'établir une nouvelle session de manière abrégée et de réduire l'utilisation du CPU en réutilisant les paramètres cryptographiques de la session précédente.

Le ***TLS session resumption***, également appelé "*TLS session caching*" (RFC 5077[96]), est un mécanisme qui optimise les performances de TLS en sacrifiant légèrement sa sécurité. Lors de l'établissement d'une

[95] https://www.imperialviolet.org/2012/04/11/falsestart.html

[96] https://tools.ietf.org/html/rfc5077

session TLS, le client et le serveur négocient et dérivent une clé maîtresse, une opération coûteuse sur le plan cryptographique. Si le *TLS session caching* est utilisé, les connexions ultérieures qui font partie de la même session utiliseront la même clé principale, ce qui réduit le temps de connexion (à 1-RTT). Si un attaquant potentiel obtient la clé maîtresse, il pourrait déchiffrer toutes les connexions associées à la même session. Toutefois, comme les sessions TLS ne doivent pas durer dans le temps, leur utilisation est acceptable du point de vue de la sécurité, compte tenu des avantages obtenus en termes de rendement.

Il est recommandé de définir la période maximale pendant laquelle les sessions TLS seront mises en cache, en utilisant par exemple un jour (24 heures) comme référence. Il est également recommandé de ne pas partager le cache de session TLS entre différents environnements, serveurs ou applications Web (dans les environnements Web d'hébergement partagé).

Les sessions sont identifiées à la fois sur le client et sur le serveur par un identifiant de session (Session ID - RFC 5246[55]), échangé entre les deux parties lors de l'établissement initial de la session. Lors de l'utilisation du *TLS session caching*, le client et le serveur doivent conserver les informations relatives à l'état de la session pendant un certain temps.

L'alternative (pour éviter le stockage côté serveur, en particulier lorsque le serveur gère des milliers ou des millions de clients Web) est d'utiliser des tickets de session (*TLS session tickets* - RFC 507796), où tout l'état de la session est conservé sur le client (également connu sous le nom de *Stateless TLS Session Resumption*). Du point de vue de la sécurité, l'utilisation de *TLS session tickets* est déconseillée, car ils exposent tous les détails cryptographiques de l'état de la session sur le canal de communication, protégés uniquement par une clé de ticket qui, dans certains cas, peut être plus faible que les clés de session elles-mêmes. Par exemple, OpenSSL utilise AES 128 bits pour les clés de ticket et réutilise par défaut les clés de ticket entre les sessions, qui sont créées au démarrage du serveur Web et ne font pas l'objet d'une rotation (elles peuvent être utilisées pendant de très longues périodes en réduisant à néant l'efficacité du *Forward Secrecy*). De même, dans le cas de l'utilisation du *TLS session tickets*, il est recommandé de ne pas partager la clé du ticket entre différents environnements, serveurs ou applications Web (dans les environnements Web d'hébergement partagé).

Par exemple, Apache 2.4.12+ possède des capacités avancées pour la gestion du cache de session TLS, la désactivation des tickets de session et la gestion des clés de ticket.

3.7.2.3 *PRECONNECT*

La balise HTML <link>[97] spécifie les relations entre le document Web actuel et d'autres ressources externes, telles que les feuilles de style (CSS ou *stylesheets*), mais peut également être utilisée pour référencer d'autres ressources Web (p. ex. externes)

La directive HTML "preconnect"[98] [Réf. - 26] de <link> (<link rel="preconnect" href="https://...">) permet d'indiquer au navigateur ou au client Web d'ouvrir prématurément une connexion HTTPS au serveur et au port spécifiés pour charger des ressources qui seront référencées ultérieurement. Ces pré-indications améliorent le rendement en effectuant à l'avance la résolution DNS, l'établissement de la connexion TCP et la négociation TLS. La directive "preconnect" est prise en charge par Chrome 46[99], Firefox 39[100], et Opera 33[101], et complète d'autres directives Web visant à améliorer le rendement, telles que "preload", "prefetch" ou "prerender"[102].

3.7.3 Forcer l'utilisation (par défaut) de HTTPS

L'un des problèmes qui sous-tend encore les technologies Web est que le protocole HTTP est toujours utilisé par défaut, et non le protocole HTTPS, qui est une fonction facultative. Afin de mettre en œuvre un serveur ou une application Web qui soit uniquement disponible sur HTTPS, il est recommandé de rediriger automatiquement tout le trafic HTTP (TCP/80) vers HTTPS (TCP/443 ; voir section "3.1. Accès aux ports TCP/IP associés aux services Web").

En aucun cas, un utilisateur accédant à la version HTTPS de l'environnement, du serveur ou de l'application Web ne doit être redirigé vers la version HTTP, qui ne fait pas appel aux mécanismes d'authentification et/ou de chiffrement.

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/

3.7.4 Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS

L'en-tête de sécurité du protocole HTTP connu sous le nom de HSTS ou STS (*HTTP Strict Transport Security*) [Réf. - 27] permet au serveur Web de déclarer (et d'informer les clients Web) qu'il n'est accessible que via HTTPS (et non HTTP). Par conséquent, les navigateurs et clients Web prenant en charge HSTS ne génèrent aucune requête HTTP vers l'environnement Web et utilisent automatiquement le protocole HTTPS à tout moment.

HSTS empêche les erreurs de configuration et de mise en œuvre, ainsi que les attaques MitM passives, basées sur la capture du trafic non chiffré, et les attaques actives, connues sous le nom de SSLstrip, c'est-à-dire lorsqu'un attaquant potentiel force le client Web victime à se connecter au serveur Web en utilisant HTTP, sans utiliser de chiffrement, au lieu de HTTPS et, par conséquent, en révélant des informations sensibles telles que des identifiants, des cookies ou des ID de session, etc.

En outre, l'utilisation du HSTS atténue les attaques puisqu'elle évite la possibilité que l'utilisateur accepte des certificats numériques invalides. Les alertes de sécurité et les erreurs dans la validation des certificats numériques dans les navigateurs Web ne peuvent pas être ignorées ou évitées par les utilisateurs lorsqu'ils utilisent HSTS.

Les alertes de sécurité et les erreurs dans la validation des certificats numériques dans les navigateurs Web ne peuvent pas être ignorées ou évitées par les utilisateurs lorsqu'ils utilisent HSTS

L'en-tête HSTS doit être inclus dans chaque réponse HTTPS générée par le serveur ou l'application Web :

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

Il convient de noter que le navigateur Web impose l'utilisation du protocole HTTPS pour un serveur Web particulier pendant la période indiquée par la directive "max-age" (spécifiée en secondes) dans l'en-tête HSTS.

Il est recommandé que la valeur "max-age" soit au moins 10886400 (18 semaines), car il s'agit de la valeur minimale requise pour être admise dans la liste préchargée (décrite ci-dessous). Toutefois, il est conseillé d'utiliser des périodes plus longues, d'au moins 6 ou 12 mois (ou 365 jours : "31536000").

Pour minimiser l'impact potentiel d'une configuration incorrecte lors du déploiement du HSTS, il est recommandé d'augmenter progressivement la valeur de "max-age", en commençant par une petite valeur (5 minutes : "300"), et en l'augmentant jusqu'à des valeurs plus importantes (1 semaine : "604800", 1 mois : "2592000", et enfin jusqu'à sa valeur finale). En même temps, les journaux d'accès du serveur Web doivent être surveillés de près pour détecter les requêtes reçues par HTTP (et non HTTPS).

Du point de vue de la sécurité, le HSTS doit être mis en œuvre pour l'ensemble du domaine, et pas seulement pour les serveurs Web individuels. Pour cela, il faut utiliser la directive "includeSubDomains". Pour éviter les problèmes d'accessibilité, tous les serveurs et applications Web du domaine doivent utiliser le protocole HTTPS et disposer de certificats numériques valides. L'application du HSTS à tous les sous-domaines atténue les attaques de DNS et la génération de nouveaux noms de serveurs Web qui n'existent pas en réalité et sur lesquels les connexions HTTPS ne seraient pas utilisées (comme ils n'existent pas, il n'y a pas d'en-tête HSTS ou de directive spécifique pour eux dans la liste préchargée).

Du point de vue de la sécurité, le HSTS doit être mis en œuvre pour l'ensemble du domaine, et pas seulement pour les serveurs Web individuels

En outre, il est important de noter que HSTS est un mécanisme de sécurité TOFU (Trust On First Use), c'est-à-dire qu'il ne s'applique pas la première fois que le client Web se connecte au serveur Web. Pour pallier cette faiblesse, la directive "preload" indique l'intérêt et la possibilité que le serveur Web ou le domaine associé ait été inclus dans la liste de préchargement HSTS des navigateurs Web, "HSTS Preload List". En réalité, la liste préchargée ne couvre que des domaines entiers, et il n'est pas possible d'ajouter des serveurs Web individuels.

La liste HSTS préchargée [Réf. - 28] est une liste interne aux navigateurs Web, gérée par Chrome/Google, qui est utilisée comme référence par tous les autres navigateurs Web pour déterminer quels domaines veulent faire un usage exclusif du HTTPS et donc pour lesquels le navigateur Web ne générera jamais de trafic HTTP (évitant ainsi les attaques dès la première connexion du client Web).

Pour qu'un domaine soit admis dans la liste préchargée, il doit répondre à des exigences minimales [Réf. - 28][103] (à partir de février 2016) similaires à celles décrites ci-dessus, et les maintenir à partir du moment où il est inclus dans la liste. Il est possible de vérifier la liste complète préchargée [Réf. - 29], y compris tous les domaines existants sur la

[103] https://hstspreload.org/#submission-requirements

liste, ainsi que la liste en attente [Réf. - 30] qui sera incluse dans les futures versions des navigateurs Web. Il est recommandé de surveiller ces listes, en particulier la dernière, une fois que la requête pour qu'un domaine soit inclus dans la liste.

En résumé, l'activation de HSTS doit être considérée comme une déclaration prévisionnelle du serveur Web et, de préférence, du propriétaire du domaine, selon laquelle HTTPS sera pris en charge pendant toute la durée de vie du domaine (*. dominio.es). Une fois que le domaine a été ajouté à la liste préchargée, il n'est pas facile de le supprimer[104].

3.7.5 Éviter le contenu mixte HTTP sur HTTPS

L'objectif de la mise en œuvre du protocole HTTPS que nous proposons dans notre rapport est que tous les environnements, serveurs ou applications Web utilisent le protocole HTTPS pour l'ensemble de leurs contenus, sans exception.

L'objectif de la mise en œuvre du protocole HTTPS que nous proposons dans notre rapport est que tous les environnements, serveurs ou applications Web utilisent le protocole HTTPS pour l'ensemble de leurs contenus, sans exception

Les pages Web de contenu mixte [Réf. - 31] (ou mélangé) sont des pages servies initialement en HTTPS mais qui contiennent des éléments chargés (références de contenu et de ressources Web) en utilisant le protocole HTTP en clair (c'est-à-dire une connexion en texte brut non-chiffrée et sans authentification), comme des images, d'autres pages Web, des *frames* ou des *iframes*, des scripts Javascript, des feuilles de style (CSS), etc. Malheureusement, une seule référence HTPP pourrait être interceptée et manipulée par un attaquant potentiel pour compromettre la session HTTPS de l'utilisateur.

On définit généralement deux catégories pour le contenu mixte : le contenu passif ou moins pertinent (car il ne peut pas interagir avec le reste de la page, comme les images et les contenus multimédias) et le contenu actif ou critique (en raison de sa capacité à interagir avec la page Web entière), comme les scripts, les contenus et les balises HTML (*iframes*), les feuilles de style (CSS), les objets Flash ou Java, etc. L'inclusion de contenu actif permettrait à un attaquant potentiel de prendre le contrôle total de la session et de pouvoir effectuer n'importe quelle action dans l'environnement Web en se faisant passer pour l'utilisateur victime. La plupart du temps, ce contenu actif mixte est bloqué par les navigateurs Web[105]. Toutefois, même avec du contenu mixte

[104] https://hstspreload.org/#removal

[105] En fonction du type de navigateur Web utilisé et de sa version.

passif, il serait possible de manipuler l'utilisateur à partir du contenu des images qu'il consulte (par le biais de l'ingénierie sociale), d'envoyer des images contenant des *exploits* ou d'effectuer des attaques par *content sniffing* (où une image pourrait être interprétée par le navigateur ou le client Web comme un script), de compromettre la confidentialité et de suivre l'activité de l'utilisateur.

Il est recommandé de revoir l'ensemble du contenu et du code associés à l'environnement, à l'application et aux pages Web (statiques et dynamiques) et de remplacer toutes les références à "http://" par "https://" (ou par des références relatives ou absolues sans spécifier le protocole), à l'exception des liens externes vers d'autres pages Web (dans l'attribut "href" de la balise d'ancrage "<a>") qui, à quelques exceptions près, ne sont pas considérés comme du contenu mixte (car ils redirigent le navigateur Web vers une autre page Web). Il convient donc d'établir une distinction entre les liens Web (vers des sites Web externes échappant au contrôle de l'environnement Web) et l'inclusion de ressources Web, intégrées dans l'environnement Web lui-même (qu'elles soient internes ou externes). Les contenus qui sont chargés en tant que portion de la page Web, tels que les feuilles de style (CSS), les images, les scripts, le contenu HTML (*iframes*), etc.[106] ont une importance spéciale. Le processus de conversion de tout le contenu Web en HTTPS peut être complexe et fastidieux en fonction de la complexité et de la longueur du site Web, du contenu actuel et d'autres dépendances éventuelles[107].

Il est recommandé de revoir l'ensemble du contenu et du code associés à l'environnement, à l'application et aux pages Web et de remplacer toutes les références à "http://" par "https://", à l'exception des liens externes vers d'autres pages Web qui, à quelques exceptions près, ne sont pas considérés comme du contenu mixte

Outre les références propres, c'est-à-dire celles qui sont associées au même environnement Web, il convient d'accorder une attention particulière aux références de ressources externes appartenant à des services tiers (telles que les bibliothèques, les services publicitaires, les cartes, l'intégration aux réseaux sociaux ou toute autre fonctionnalité), car leur contrôle est plus limité en faveur de l'usage exclusif du HTTPS. Toutes les ressources tierces doivent être référencées par des liens HTTPS, et leur validité et leur intégrité (contre les manipulations inattendues) doivent être vérifiées par une nouvelle norme appelée *SubResource Integrity* (SRI)[108].

Il est très important de savoir que l'inclusion de ressources et de composants de services Web tiers dans votre propre page Web crée un lien de confiance très étroit entre les différents sites Web. Par exemple, l'inclusion d'un code Javascript tiers dans l'application Web fait que ce code soit considéré comme un code hébergé localement, car toute action

[106] hhttps://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/

malveillante, vulnérabilité ou manipulation éventuelle du code aurait un contrôle total sur l'application Web et un effet négatif direct sur celle-ci.

De même, il se peut que les navigateurs ou clients Web n'autorisent pas l'utilisation ou le chargement de contenu mixte (en particulier, le contenu actif) et que chacun impose des critères et des restrictions différents à cet égard. Ceux-ci pourront être évalués par le biais du service "Test client SSL" [Réf. - 3] ("Mixed Content Handling").

Il est également possible d'évaluer l'existence de contenu mixte dans un environnement Web HTTPS à l'aide d'outils tels que le Mixed Content Scan[109].

L'objectif consistant à éviter le contenu mixte HTTP sur HTTPS peut être renforcé et simplifié au moyen de CSP (voir section "3.7.6. *Content Security Policy* (CSP)"), surtout pour les environnements Web plus grands et plus complexes, et par l'utilisation de HSTS (voir section "3.7.4. Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS").

3.7.6 *Content Security Policy* (CSP)

La politique Web standardisée connue sous le nom de *Content Security Policy* (CSP) [Réf. - 32], outre ses multiples usages pour atténuer d'autres types d'attaques associées aux technologies Web (p. ex. *Cross-Site Scripting* ou XSS), peut être utilisée pour améliorer la mise en œuvre de HTTPS.

La politique Web standardisée connue sous le nom de *Content Security Policy* (CSP) [Réf. - 32], outre ses multiples usages pour atténuer d'autres types d'attaques associées aux technologies Web, peut être utilisée pour améliorer la mise en oeuvre de HTTPS

La méthode CSP est un mécanisme de sécurité qui permet de restreindre le comportement des navigateurs et clients Web, en fournissant une méthode standard aux propriétaires de sites Web pour qu'ils puissent contrôler la façon dont les différentes ressources d'une page Web sont obtenues (et chargées) et déclarer les origines approuvées de contenu dont les navigateurs devraient être autorisés à charger sur ce site. L'en-tête HTTP "Content-Security-Policy"[110]. est utilisé pour sa mise en œuvre. Afin de garantir la protection des environnements Web, il est recommandé d'utiliser la méthode CSP[111] d'une manière beaucoup plus large et rigoureuse que celle détaillée ci-dessous, qui se concentre uniquement sur les aspects liés à l'utilisation de HTTPS. L'outil "CSP Builder"[112] peut être utilisé pour créer la stratégie CSP, tandis que "CSP Analyser"[113] peut être utilisé pour analyser une méthode CSP existante.

[109] https://github.com/bramus/mixed-content-scan

[110] https://scotthelme.co.uk/content-security-policy-an-introduction/

[111] https://scotthelme.co.uk/csp-cheat-sheet/

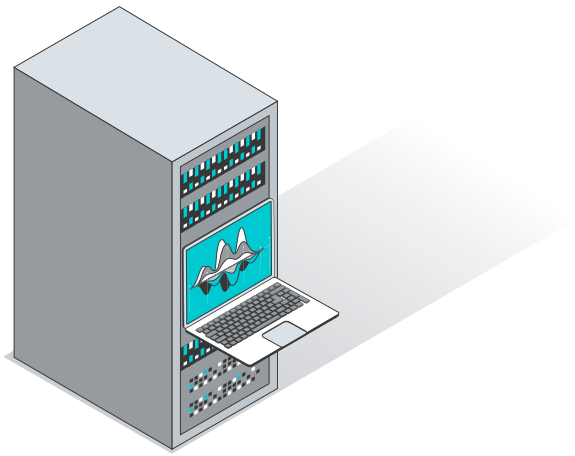
[112] https://report-uri.io/home/generate

[113] https://report-uri.io/home/analyse

La politique CSP a beaucoup évolué au cours des dernières années. Identifiée initialement par la version 1.0 de la norme [Réf. - 33], elle est prise en charge par la plupart des navigateurs Web actuels[114] (sauf IE[115]). Par la suite, la version 2 de CSP ou "niveau 2" [Réf. - 34], a ajouté de nouvelles directives, de nouvelles fonctionnalités et de nouveaux mécanismes de protection, dont certains sont associés au HTTPS et seront décrits plus loin. La prise en charge de toutes les fonctionnalités du niveau 2 de CSP par les navigateurs Web modernes est assez répandue, mais plus limitée[114]. Actuellement, la version 3 de CSP ou "niveau 3" [Réf. - 35] est en cours de révision et de conception, y compris les nouvelles directives et les modifications apportées aux versions précédentes.

Dans le cas de HTTPS, CSP peut être utilisée pour empêcher la récupération de ressources via des liens HTTP qui n'utilisent pas HTTPS, empêchant ainsi l'utilisation de contenu mixte (voir section "3.7.5. Éviter le contenu mixte HTTP sur HTTPS").

D'une part, la directive "default-src" de CSP peut être utilisée pour indiquer au navigateur Web d'où il est autorisé à charger les ressources (y compris les images, les scripts, les CSS, etc.) par défaut associées au serveur ou à l'application Web. Si sa valeur est "https :", cela indique qu'ils ne peuvent être récupérés que via HTTPS, et non HTTP, quel que soit le domaine à partir duquel ils sont récupérés (cette simple stratégie CSP n'applique aucune restriction à cet égard). Cette stratégie peut également s'étendre aux formulaires Web via la directive "form-action" avec la même valeur.



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

REMARQUE :
Dans le cas de CSP, comme pour HPKP, il est possible d'appliquer strictement et efficacement la politique tout en recevant des rapports de violation de celle-ci (via "report-uri"). Cette option (décrite plus loin à côté de "...-Report-Only") est très intéressante pour identifier les ressources qui ne devraient pas être référencées, ou les attaques qui ont inclus de nouveaux contenus Web non autorisés.

[114] http://caniuse.com/#search=csp (http://caniuse.com/#feat=contentsecuritypolicy et http://caniuse.com/#feat=contentsecuritypolicy2)

[115] IE utilise l'ancien en-tête HTTP "X-Content-Security-Policy".

Grâce à la politique CSP, le navigateur Web ne chargera aucune ressource ou ne soumettra aucun formulaire qui n'utilise pas le protocole HTTPS. La directive "default-src" spécifie la politique de sécurité à appliquer par défaut, c'est-à-dire pour tous les types de ressources qui n'ont pas de directives définissant une politique CSP spécifique pour les autres contenus (sources, images et *favicons*, multimédia–audio et vidéo, éléments enfants–comme les *frames* et les *iframes*, objets, scripts–Javascript, styles, connexions–comme AJAX (XMLHttpRequest) et WebSockets, etc.).

Grâce à la politique CSP, le navigateur Web ne chargera aucune ressource ou ne soumettra aucun formulaire qui n'utilise pas le protocole HTTPS

De plus, comme pour les autres en-têtes HTTP de sécurité tels que HPKP (voir section "3.7.4. Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS"), et toujours pour minimiser l'impact potentiel d'une mauvaise configuration lors du déploiement de CSP, il est recommandé de commencer par utiliser les capacités de rapport de CSP via l'en-tête étendu "...-Report-Only" et la directive "report-uri". Il est recommandé de tester initialement cet en-tête pour identifier le bon fonctionnement de la politique. Si un navigateur ou un client Web reçoit CSP Report-Only[116] et que le contenu Web fait référence à une ressource qui n'utilise pas le protocole HTTPS, il ne bloquera pas le trafic HTTPS (comme avec l'en-tête CSP standard, qui applique la politique CSP sans exception), mais générera un rapport dirigé vers l'URL affichée dans cet en-tête par la directive "report-uri". Dans ce scénario, les clients Web sont utilisés efficacement comme capteurs pour détecter les accès HTTP aux ressources :

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

Contrairement à HPKP, dans le cas de CSP il est possible d'utiliser une référence Web HTTPS associée au même serveur Web qui a envoyé l'en-tête CSP (ou à un autre) pour la directive "report-uri" puisque, dans ce cas, le navigateur Web n'aura aucun problème pour envoyer le rapport (à différence de ce qui peut arriver dans HPKP).

Dans le cas de CSP, il est possible d'utiliser simultanément les en-têtes "Content-Security-Policy" et "Content-Security-Policy-Report-Only". La première permettrait d'appliquer efficacement la politique CSP actuelle, et la seconde de vérifier les éventuelles violations qu'une nouvelle politique CSP (introduite dans l'environnement Web) pourrait provoquer.

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

Il est recommandé de commencer à utiliser l'en-tête CSP sur une ressource Web spécifique (ou un petit ensemble de ressources Web), jusqu'à ce que vous vérifiiez sa bonne configuration, afin d'éviter de recevoir une multitude de rapports de violation de CSP. Contrairement à HSTS, CSP est une politique qui est appliquée individuellement pour chaque page Web (et n'est pas mise en cache par le navigateur Web), donc chaque ressource Web à protéger doit inclure l'en-tête CSP dans sa réponse. L'impact d'une mauvaise configuration de la politique est minime, car elle ne compromettrait chaque ressource que de manière individuelle et en une seule fois.

Il est recommandé de commencer à utiliser l'en-tête CSP sur une ressource Web spécifique, jusqu'à ce que vous vérifiiez sa bonne configuration, afin d'éviter de recevoir une multitude de rapports de violation de CSP

L'objectif ultime de la migration vers HTTPS est la modification de tous les contenus de l'environnement, du serveur ou de l'application Web afin que toutes les références fassent usage de HTTPS au lieu de HTTP. Toutefois, ce processus de migration peut s'avérer complexe, car il est nécessaire de mettre à jour toutes les références, tant statiques (contenues dans des fichiers) que dynamiques (contenues dans des bases de données) vers "https://". Il est recommandé d'utiliser des références relatives (telles que "./répertoire/resource.html") et absolues (telles que "/répertoire/resource.html"), ou au moins d'utiliser des références qui ne spécifient pas le protocole (telles que "//domaine.es/directoire/resource.html"), afin de généraliser l'utilisation de HTTPS (et d'éviter les références directes vers HTTP). L'utilisation des capacités de notification de CSP mentionnées ci-dessus peut être cruciale pour identifier les ressources qui utilisent encore le protocole HTTP ("http://")[117].

De plus, pendant le processus de migration (et après), la directive "upgrade-insecure-requests" [Réf. - 36] (associée à CSP [Réf. - 33]) peut être d'une grande aide. En utilisant "upgrade-insecure-requests", l'environnement Web demandera aux navigateurs Web qui prennent en charge cette directive[118] (tels que Firefox 42, Chrome 43 ou Opera 30) de mettre à jour toutes les références HTTP vers HTTPS, aussi bien pour le contenu de l'environnement Web que pour celui de tiers, avant d'envoyer la requête associée. Le seul problème qui peut survenir lors de l'utilisation de cette politique est que les ressources qui ne sont pas disponibles via HTTPS ne peuvent pas être chargées. Les navigateurs qui prennent en charge cette directive l'indiquent au moyen de l'en-tête "Upgrade-Insecure-Requests : 1" dans leurs requêtes HTTP.

[117] https://scotthelme.co.uk/fixing-mixed-content-with-csp/

[118] http://caniuse.com/#feat=upgradeinsecurerequests

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

Une autre directive associée à "upgrade-insecure-requests", mais plus restrictive, est "block-all-mixed-content" [Réf. - 31][119] (associée à CSP 3 [Réf. - 35]). En utilisant "block-all-mixed-content", l'environnement Web indiquera aux navigateurs Web qui supportent cette directive de ne pas charger de contenu via HTTP si la page Web a été récupérée via HTTPS. Cette directive n'autorise aucune requête via HTTP. Dans le cas de l'utilisation de "upgrade-insecure-requests" avec "block-all-mixed-content", cette dernière directive ne s'appliquera pas, et le navigateur Web sera plus permissif par rapport au chargement du contenu via HTTP, et essaiera au moins de l'obtenir via HTTPS.

En utilisant "block-all-mixed-content", l'environnement Web indiquera aux navigateurs Web qui supportent cette directive de ne pas charger de contenu via HTTP si la page Web a été récupérée via HTTPS

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

REMARQUE :
La présence de plusieurs en-têtes CSP dans la même réponse HTTP fait que le navigateur Web combine toutes les politiques reçues, en appliquant de manière restrictive le plus petit commun multiple de toutes ces politiques[106].

En résumé, pour donner un exemple d'une politique simple axée sur HTTPS, CSP peut être utilisé pour restreindre l'utilisation de contenu mixte tiers, en forçant l'utilisation de HTTPS, un scénario où HSTS ne peut pas imposer de restrictions (voir section "3.7.4. Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS"), en utilisant la politique de sécurité suivante :

```
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"
```

REMARQUE :
L'exemple de politique ci-dessus[120] précise que toute référence à AJAX (XMLHttpRequest), fetch, EventSource et, surtout, WebSockets doit également faire appel à HTTPS ou WSS (WebSockets Secure) via "connect-src".

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

120 L'exemple de politique présenté pourrait être plus strict, et interdire le chargement et l'utilisation de contenu propre via HTTPS catalogué comme "non sûr", c'est-à-dire pour des ressources telles que les scripts (<script>) et les styles (<style>) définis dans la page Web elle-même ("unsafe-inline"), ainsi que les appels Javascript à "eval ()" et similaires ("unsafe-eval"), en supprimant les directives 'unsafe-inline' et 'unsafe-eval' de "default-src https" : Ces directives ne sont pas directement liées à HTTPS.

3.7.7 Manque de fonctionnalités avancées dans les environnements HTTP

Certains navigateurs et clients Web évaluent la possibilité et mettent en œuvre des mesures visant à interdire l'accès aux fonctionnalités avancées disponibles dans le navigateur pour les sites Web basés sur le protocole HTTP, qui ne peuvent être utilisées que via le protocole HTTPS (qui fait partie des "origines sûres") [Réf. - 37].

Parmi la liste des fonctionnalités avancées proposées par Chrome figurent celles liées à la géolocalisation, à l'orientation et au déplacement de l'appareil de l'utilisateur, à l'accès à la caméra, aux notifications, etc.

3.7.8 Éviter la mise en cache de contenus sensibles

L'objectif de ce rapport est que tout le trafic associé aux technologies Web utilise le protocole HTTPS, et il est donc nécessaire de faire la distinction entre le contenu sensible et le contenu public. D'une manière générale, il est recommandé de ne pas stocker localement sur le disque, ou en cache, sur les clients Web, le contenu sensible échangé par l'environnement, le serveur ou l'application Web via HTTPS.

L'objectif de ce rapport est que tout le trafic associé aux technologies Web utilise le protocole HTTPS, et il est donc nécessaire de faire la distinction entre le contenu sensible et le contenu public

Pour ce faire, il est nécessaire d'utiliser différents en-têtes HTTP (avec des directives de mise en cache) qui indiquent aux navigateurs et clients Web standard de ne pas stocker une copie locale, ou activer une mise en cache, de certains contenus critiques[121]. Les en-têtes HTTP à utiliser varient en fonction des navigateurs et clients Web utilisés pour accéder au contenu Web (car tous ne déploient pas correctement les standards[122]). Il est donc recommandé de spécifier toutes les directives suivantes sur toutes les ressources dont le contenu est sensible et qui sont échangées via HTTPS :

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

Les directives pour le contrôle du cache du contenu Web sont actuellement définies dans la norme RFC 7234 [Réf. - 38].

[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

En revanche, les ressources et les contenus publics doivent être correctement classés afin qu'ils puissent être mis en cache même en utilisant HTTPS :

```
Cache-Control: public
```

3.7.9 Inspection du trafic HTTPS

Bien qu'il soit recommandé, d'une façon globale, d'utiliser exclusivement le protocole HTTPS pour l'échange de contenu Web, il pourrait exister des scénarios où il serait nécessaire d'autoriser l'inspection ou la surveillance du trafic Web, par exemple pour détecter la présence éventuelle de trafic ou de contenu malveillant, ou d'attaques qui pourraient être dissimulées par le trafic chiffré.

L'inspection (ou l'interception) du trafic HTTPS ou TLS est toujours controversée, avec des arguments pour et contre (en particulier aujourd'hui, où on observe une tendance généralisée à une plus large utilisation de ces protocoles). En général, cette capacité est requise par les solutions de surveillance du trafic (telles que les systèmes de détection d'intrusion, IDS) et les réseaux de communication, en particulier, dans les réseaux internes et privés des organisations, ou dans les infrastructures périmétriques se connectant à Internet. Sa mise en œuvre peut se faire à partir du réseau lui-même (à l'aide de proxies d'interception) et/ou dans les navigateurs ou clients Web, selon le cas.

Parmi les scénarios spécifiques où l'inspection du trafic HTTPS peut être positive d'un point de vue défensif, citons l'identification des téléchargements de logiciels malveillants, la détection du trafic dirigé vers les serveurs de commande et contrôle (C&C) d'un *botnet*, les scénarios d'exfiltration de données d'une organisation, etc. Sans les capacités nécessaires pour inspecter le contenu des connexions HTTPS, il est seulement possible d'obtenir des détails sur les métadonnées de la connexion, c'est-à-dire les adresses IP concernées, les noms dans le DNS et les différents services utilisés (s'ils sont autres que TCP/443), ou d'obtenir certains modèles de trafic. Dans tous les cas, les solutions d'inspection doivent être totalement transparentes pour l'utilisateur et ne doivent pas affecter les autres recommandations de sensibilisation et les bonnes pratiques qui lui ont été transmises pour identifier les attaques réelles sur HTTPS[123].

Bien qu'il soit recommandé, d'une façon globale, d'utiliser exclusivement le protocole HTTPS pour l'échange de contenu Web, il pourrait exister des scénarios où il serait nécessaire d'autoriser l'inspection ou la surveillance du trafic Web, par exemple pour détecter la présence éventuelle de trafic ou de contenu malveillant, ou d'attaques qui pourraient être dissimulées par le trafic chiffré

[123] https://www.us-cert.gov/ncas/alerts/TA17-075A

Malheureusement, l'utilisation des solutions d'inspection du trafic HTTPS est très répandue (entre 4 et 11 % des connexions HTTPS publiques sur Internet) et, dans de nombreux cas, leur mise en œuvre diminue le niveau de sécurité des communications HTTPS (dans 62 % des cas), par exemple en utilisant des algorithmes ou des suites cryptographiques faibles, ou introduit même des vulnérabilités de sécurité supplémentaires (dans 58 % des cas), par exemple en validant incorrectement les certificats numériques[124]. Si la mise en œuvre d'une telle solution est en cours d'évaluation, il est recommandé d'évaluer soigneusement ses avantages et ses inconvénients. En cas d'utilisation d'une solution d'inspection du trafic HTTPS, il est recommandé d'évaluer au moins partiellement le comportement de cette solution en utilisant le service "BadSSL" [Réf. - 20].

L'utilisation des solutions d'inspection du trafic HTTPS est très répandue et, dans de nombreux cas, leur mise en œuvre diminue le niveau de sécurité des communications HTTPS

D'autre part, l'inspection du trafic HTTPS est, par nature, contraire à l'ensemble des recommandations fournies tout au long de ce rapport, rendant inutiles les différents mécanismes et capacités de protection suggérés et facilitant la possibilité d'accéder au trafic sensible qui devrait être protégé (à des fins d'inspection et/ou de manipulation) et d'usurper l'identité des points d'extrémité de communication[124]. Les solutions d'inspection du trafic réalisent une attaque de type Man-in-the-Middle (MitM) d'une façon technique et efficace (même avec autorisation, et quel que soit le trafic échangé : personnel, professionnel, etc.), alors que l'on tente en fait d'atténuer cette menace par l'utilisation du HTTPS.

Vu le niveau de sophistication et les progrès actuels des mécanismes de protection associés à HTTPS, les solutions d'inspection nécessitent non seulement des modifications de l'architecture du réseau, mais aussi de la gestion des connexions HTTPS entre deux points (tant du côté client que du côté serveur), faute de quoi leur activation deviendra difficile (p. ex. si l'on utilise CT ou HPKP ; dans ce dernier cas, à moins que les AC ne soient manipulées localement dans les navigateurs et clients Web, aussi bien d'entreprise que personnels).

Enfin, même au cours du processus de définition et de ratification de la norme TLS 1.3 [Réf. - 22], différents groupes (p. ex. du secteur financier[126]) ont exprimé des inquiétudes quant à la suppression de l'échange de clés à l'aide de RSA (autorisant uniquement le DHE et l'ECDHE en faisant usage du Forward Secrecy) dans TLS 1.3, ce qui élimine la possibilité d'inspecter et de déchiffrer le trafic TLS dans les réseaux internes

[124] https://jhalderm.com/pub/papers/interception-ndss17.pdf

[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

et les centres de données. C'est pourquoi une nouvelle norme a même été proposée pour permettre aux serveurs HTTPS d'être configurés de manière à utiliser une clé DH ou ECDH statique, plutôt qu'une clé éphémère, pour toutes les connexions TLS, ce qui permettrait de surveiller le trafic associé à ces serveurs[127].

3.7.10 Cookies sécurisés

Indépendamment de si l'on utilise d'autres mécanismes de sécurité décrits dans le présent rapport, tels que le HSTS (voir section "3.7.4. Forcer l'utilisation stricte (par défaut) de HTTPS : HSTS"), il est recommandé de définir la directive "Secure" sur tous les cookies utilisés par l'application Web afin qu'ils ne soient envoyés par les navigateurs et clients Web que par une connexion HTTPS, et jamais par HTTP.

Même si l'on utilise HSTS, il existe des scénarios où il pourrait être possible d'obtenir la valeur d'un cookie si vous n'avez pas la directive "Secure", par exemple lors de la première connexion à l'environnement Web, ou lors de l'établissement de connexions à d'autres serveurs du même domaine. L'utilisation de la directive HSTS "includeSubDomains" est particulièrement pertinente lors de l'utilisation de cookies de domaine, dont la portée et l'application incluent l'ensemble du domaine (et non pas seulement l'application Web qui a défini la valeur du cookie).

Il est recommandé d'utiliser les autres directives de protection des cookies[128], même si elles ne sont pas directement liées à HTTPS (p. ex. HttpOnly ou Same-site cookies[129]). Dans le cas de la proposition associée aux préfixes des cookies[130], le préfixe "__Secure-" et le préfixe "__Host-" exigent que le cookie comporte la directive "Secure" et qu'il ait été servi via HTTPS.

Indépendamment de si l'on utilise d'autres mécanismes de sécurité décrits dans le présent rapport, tels que le HSTS, il est recommandé de définir la directive "Secure" sur tous les cookies utilisés par l'application Web afin qu'ils ne soient envoyés par les navigateurs et clients Web que par une connexion HTTPS, et jamais par HTTP

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 Recommandations relatives au contenu et aux applications Web HTTPS

Le résumé des recommandations relatives au contenu et aux applications Web HTTPS comprend :

- L'utilisation de HTTPS permet aux sites Web d'avoir plus de pertinence dans le positionnement (ou classement) des moteurs de recherche, tels que Google.
- Les moteurs de recherche donnent priorité, par défaut, à l'indexation des pages Web en utilisant la version HTTPS des sites Web. Il est donc recommandé de ne pas ajouter la directive "noindex" pour les URL qui utilisent le protocole HTTPS.
- La politique définie dans le fichier "robots.txt" doit permettre l'indexation du contenu HTTPS du serveur Web par les moteurs de recherche.
- Il est recommandé d'utiliser la balise "rel=canonical" pour définir que les ressources canoniques ou de référence au sein du serveur Web sont celles qui utilisent le protocole HTTPS (qu'elles soient uniques ou dupliquées).
- Évaluez en détail la mise en œuvre de l'amélioration des performances de TLS, telles que le *TLS handshake* (1-RTT), le *TLS record size* et la chaîne de certification, le *TLS session resumption* ou *TLS session caching* (*TLS session tickets*), le *TLS False Start*, le *TCP Fast Open*, le *TLS early termination*, l'OCSP, etc.
- Utilisez les mécanismes de compression Brotli via HTTPS.
- Utilisez le système *TLS False Start* avec les mécanismes de sécurité qui permettent de l'utiliser en toute confiance (chiffrement 128 bits, avec *Forward Secrecy* et NPN/ALPN).
- Utilisez le *TLS session resumption* (ou *TLS session caching*) en fixant la période maximale pour laquelle les sessions TLS seront mises en cache, par exemple, à un jour (24 heures). Ne partagez pas la mise en cache des sessions TLS entre différents environnements Web d'hébergement partagé.
- N'utilisez pas de *TLS session tickets*, et si vous les utilisez, utilisez des clés de ticket robustes qui n'aient pas été réutilisées fréquemment ou partagées entre différents environnements Web d'hébergement partagé.

- Il est recommandé d'utiliser la directive "preconnect" dans la balise HTML <link> pour demander aux clients Web d'ouvrir prématurément une connexion HTTPS, afin de charger des ressources qui seront référencées ultérieurement.
- Il est recommandé de rediriger automatiquement tout le trafic HTTP vers HTTPS via une redirection 301. En aucun cas, les accès à la version HTTPS de l'environnement Web ne doivent être redirigés vers la version HTTP.
- Utilisez HSTS (*HTTP Strict Transport Security*) pour déclarer que le serveur Web est uniquement accessible via HTTPS.
- Dans le cas de l'utilisation de HSTS, il est recommandé d'utiliser une valeur "max-age" d'au moins 10886400 (18 semaines), et de préférence de 6 ou 12 mois ("31536000"). Le HSTS peut être introduit en augmentant progressivement la valeur "max-age".
- Le HSTS doit être mis en œuvre pour l'ensemble du domaine au moyen de la directive "includeSubDomains".
- La directive "preload" doit être incluse dans la configuration du HSTS et le domaine doit être transféré pour être inclus dans la liste de préchargement publique du HSTS.
- Il est recommandé de revoir tout le contenu et le code associés à l'environnement, à l'application et aux pages Web (statiques et dynamiques) et de remplacer toutes les références à "http://" par "https://" (ou par des références relatives ou absolues sans spécifier le protocole), en évitant l'utilisation de contenu mixte.
- Vérifier la validité et l'intégrité des ressources tierces à travers la *SubResource Integrity* (SRI).
- Utilisez la méthode *Content Security Policy* (CSP) pour atténuer le risque d'utilisation de contenu mixte en indiquant que toutes les ressources doivent utiliser HTTPS par défaut. Une autre solution consiste à utiliser les capacités de notification de CSP pour identifier l'utilisation des ressources Web qui continuent à utiliser HTTP.

- Il est recommandé de commencer par une utilisation limitée de l'en-tête CSP sur une ressource spécifique, et d'augmenter progressivement son utilisation sur d'autres ressources Web.
- En particulier, la politique CSP peut être améliorée par les directives "upgrade-insecure-requests" (plus permissive) et "block-all-mixed-content" (plus stricte), afin de garantir l'utilisation de HTTPS dans tout l'environnement Web.
- Il est recommandé d'utiliser des références relatives ou absolues aux ressources Web, pour autant qu'elles ne spécifient pas le protocole, afin de généraliser l'utilisation de HTTPS (et d'éviter les références directes vers HTTP).
- Il est recommandé d'empêcher les clients Web de mettre en cache localement sur le disque tout contenu sensible échangé par l'environnement Web via HTTPS, en utilisant différents en-têtes HTTP avec des directives de mise en cache.
- En revanche, les ressources et les contenus publics doivent être classés correctement afin qu'ils puissent être mis en cache (même en utilisant HTTPS).
- Dans le cas de l'utilisation d'une solution d'inspection du trafic HTTPS, il faut d'abord vérifier qu'elle soit transparente pour l'utilisateur et qu'elle n'affecte pas le reste des bonnes pratiques. Ensuite, il faut évaluer soigneusement ses avantages et ses inconvénients, en garantissant notamment sa bonne mise en œuvre pour éviter la diminution du niveau de sécurité des communications HTTPS ou l'introduction de failles de sécurité supplémentaires.
- Il est recommandé de définir la directive "Secure" sur tous les cookies utilisés par l'application Web afin que les cookies ne soient envoyés que sur une connexion HTTPS.
- Il est recommandé d'utiliser les autres directives de protection des cookies, aussi bien celles qui ne sont pas directement liées à HTTPS (p. ex. HttpOnly ou Same-site cookies) que celles qui le sont (p. ex. les préfixes "__Secure-" et "__Host-").

3.8. Vulnérabilités dans HTTPS

Cette section fournit une petite liste et une description très résumée des vulnérabilités et des attaques (terme utilisé ci-après) les plus importantes subies par les protocoles et les implémentations associés à HTTPS au cours des dernières années [Réf. - 10], référencées ou mentionnées dans ce rapport.

Certaines de ces attaques pourraient être atténuées par une simple mise à jour des composants utilisés dans l'implémentation de HTTPS (bibliothèques, serveurs Web, etc.) ou par la modification d'une configuration non sécurisée, tandis que d'autres ont entraîné des changements radicaux dans la conception du protocole TLS ou ont rendu obsolètes (et non sécurisées définitivement) des versions des protocoles et/ou des algorithmes et suites cryptographiques.

Par conséquent, il est recommandé, de manière générale, de toujours disposer de la version la plus récente (qui corrige au moins les failles de sécurité connues) des bibliothèques TLS, des serveurs et des applications Web utilisés dans la mise en œuvre de HTTPS.

Une liste des vulnérabilités et des événements pertinents de l'industrie liés à HTTPS, de 1994 à aujourd'hui, est disponible dans "SSL/TLS and PKI History" [Réf. - 39].

Les mécanismes de renégociation non sécurisés de TLS (2009) permettent les attaques MitM et les attaques par déni de service (DoS) sur le serveur Web (voir section "3.4.3. Renégociation").

BEAST (2011) est une attaque contre TLS 1.0 (et les versions précédentes de SSL) et les algorithmes de chiffrement en mode CBC (AES, DES, etc.), en raison de l'utilisation de vecteurs d'initialisation

Cette section fournit une petite liste et une description très résumée des vulnérabilités et des attaques (terme utilisé ci-après) les plus importantes subies par les protocoles et les implémentations associés à HTTPS au cours des dernières années [Réf. - 10], référencées ou mentionnées dans ce rapport

(IV) prévisibles dans les chiffrements par blocs. Il est donc recommandé d'utiliser TLS 1.1, en donnant la priorité aux suites cryptographiques basées sur des chiffrements par flux (plutôt que par blocs) tels que RC4 (bien qu'il ne soit pas recommandé actuellement en raison d'autres faiblesses découvertes en 2013 et 2015, voir RFC 7465) et, de préférence, AES-GCM avec TLS 1.2[132].

CRIME (2012) est une attaque contre les mécanismes de compression de niveau TLS qui révèle les informations associées à la session chiffrée. Par conséquent, il est recommandé de désactiver les capacités de compression TLS du serveur Web[133].

LUCKY 13 (2013) est une attaque contre les suites cryptographiques de chiffrement par blocs avec CBC, basée sur l'analyse statistique et l'identification de petites différences dans le timing des opérations de chiffrement (connues sous le nom *padding oracle attacks*). Là encore, il est recommandé d'éviter d'utiliser les suites CBC et d'utiliser plutôt les suites GCM avec TLS 1.2.

BREACH (2013) est une attaque contre les mécanismes de compression sur le dessus du protocole TLS (prolongeant l'attaque précédente CRIME). Par conséquent, il est recommandé de désactiver les capacités de compression Web HTTP standard du serveur Web (largement utilisées) et, si cela n'est pas possible, d'appliquer d'autres techniques d'atténuation plus élaborées[134], telles que *HTTP Chunked Encoding*. Les capacités de compression de Brotli peuvent également être utilisées à la place. Une autre solution consiste à désactiver les mécanismes de compression du serveur ou de l'application Web uniquement lors de la réception de requêtes provenant d'autres sites Web, sur la base de l'en-tête "Referrer" (ou si l'en-tête "Referrer" n'est pas disponible).

TIME (2013) est une attaque similaire à BREACH, également axée sur les mécanismes de compression, mais pour laquelle aucune démonstration de faisabilité ou aucun outil d'attaque n'ont été publiés.

L'attaque **triple *handshake* (2014)** permet de manipuler la session TLS en utilisant les certificats numériques du client, c'est-à-dire les capacités de renégociation (sécurisée) de TLS. En outre, cette attaque a démontré qu'il était possible pour un serveur malveillant de sélectionner

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

des paramètres DH non sécurisés qui n'étaient même pas des nombres premiers, ce qui aurait facilement brisé l'échange de clés DH. Les implémentations modernes de TLS, tant au niveau du client que du serveur, disposent de mécanismes de défense intégrés. Par conséquent, il est recommandé de mettre à jour tous les composants associés à HTTPS.

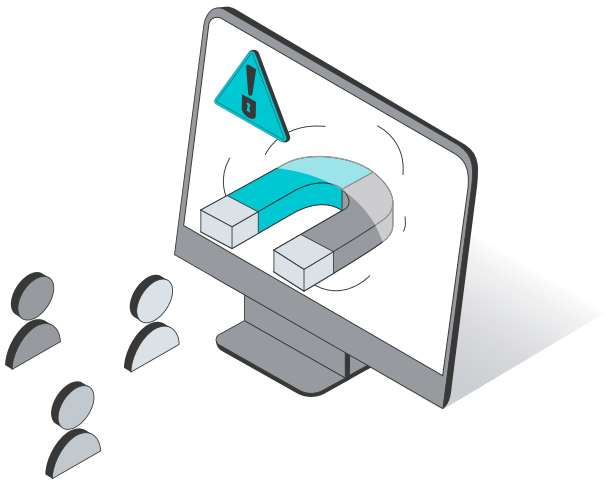
Heartbleed (2014) est une attaque spécifique à la bibliothèque OpenSSL qui permettait d'obtenir des fragments de la mémoire du serveur (par blocs de 64 KB) en manipulant la fonctionnalité *TLS Heartbeat*. Heartbleed peut être atténué en mettant à jour la version d'OpenSSL utilisée.

POODLE (2014) est une attaque contre SSL 3.0 qui permet de calculer le contenu d'une connexion chiffrée SSL (notamment avec des algorithmes de chiffrement en mode CBC ; encore une fois, en raison de la difficulté à les mettre en œuvre correctement), ainsi que de forcer une attaque en utilisant des techniques de dégradation de la version du protocole SSL/TLS, en passant de l'utilisation de TLS 1.x à l'utilisation de SSL 3.0[135] [136].

Logjam (2015) est une attaque sur les paramètres Diffie-Hellman (DH) qui donne la possibilité d'utiliser un échange de clés faible via DH[137], comme DH 768-bit ou 512-bit (ou même, pour les attaquants plus avancés, DH 1024-bit si des groupes DH connus sont utilisés). Un attaquant peut forcer, par une attaque MitM et des techniques de dégradation, l'utilisation d'une *suite* cryptographique faible qui lui permet de déchiffrer le trafic échangé. Logjam est une attaque similaire à FREAK, mais axée sur l'échange de clés utilisant DHE au lieu de RSA.

FREAK (2015) est une attaque visant les implémentations TLS des clients qui permet d'intercepter les communications HTTP et de forcer (via des techniques de dégradation) l'utilisation d'algorithmes de chiffrement faibles, par exemple RSA 512 bits. Cette attaque compromet les serveurs utilisant des suites cryptographiques qui peuvent être exportées[138].

DROWN (2016) est une attaque contre SSL 2.0 qui permet de déchiffrer le trafic HTTPS en faisant des requêtes à un serveur avec SSL 2.0 qui utilise la même clé[139]. La nouveauté de cette vulnérabilité est qu'elle



[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

permet même d'attaquer des serveurs qui ne supportent pas SSL 2.0 s'ils réutilisent la clé d'un autre serveur qui supporte cette version du protocole SSL. Bien qu'elle ait été publiée en 2016, on estime malheureusement qu'au moment de sa publication, elle avait déjà réussi à compromettre 33 % des serveurs Web HTTPS publics, du fait qu'ils assuraient encore la prise en charge de SSL 2.0.

3.8.1 Recommandations pour les vulnérabilités de HTTPS

Le résumé des recommandations pour les vulnérabilités de HTTPS, au cas où elles ne soient pas déjà mentionnées dans les sections précédentes, comprend :

- Disposez de la version la plus récente (qui corrige au moins les failles de sécurité connues) des bibliothèques TLS, des serveurs et des applications Web utilisés dans la mise en œuvre de HTTPS.
- Désactivez les capacités de compression TLS du serveur Web.
- Évitez l'utilisation de suites de chiffrement par blocs avec CBC.
- Désactivez les capacités de compression Web HTTP standard du serveur Web et, si ce n'est pas possible, appliquer d'autres techniques d'atténuation plus élaborées ou spécifiques. Les capacités de compression de Brotli peuvent également être utilisées à la place.

3.9. Compatibilité avec les navigateurs et clients Web

Bien que, du point de vue de la sécurité, il soit souhaitable de récompenser les mécanismes de sécurité et de protection les plus modernes et avancés, il faut trouver un équilibre entre le niveau de sécurité fourni par le HTTPS et sa compatibilité ou interopérabilité avec le plus grand nombre possible de clients Web.

À cette fin, avant de mettre en œuvre une nouvelle mesure de sécurité liée à HTTPS, il est recommandé d'évaluer si elle peut être prise en charge par la plupart des navigateurs ou clients Web (agents utilisateurs), tant traditionnels (p. ex. : Chrome, Edge, Firefox, IE, Opera, Safari, etc.) que mobiles (p. ex. : le navigateur Web Android (<= 4.3), Chrome (Android et iOS), Safari Mobile (iOS), Firefox mobile, etc.).

Le service Web "Can I use... ?" [Réf. - 7] permet d'évaluer quelles sont les technologies et capacités Web qui bénéficient d'une prise en charge par les navigateurs et clients Web :

Les différentes versions du protocole TLS (1.0, 1.1., 1.2, etc.) et d'autres fonctionnalités spécifiques à TLS, telles que SNI (voir section "3.4.2. SNI (*Server Name Indication*)", des directives telles que "preconnect" (voir section "3.7.2.3. *Preconnect*"), ou des suites cryptographiques spécifiques, telles que "ChaCha20-Poly1305" :	http://caniuse.com/#search=TLS
	http://caniuse.com/#search=preconnect
	http://caniuse.com/#feat=chacha20-poly1305
Prise en charge de HSTS ou STS (*HTTP Strict Transport Security*) :	http://caniuse.com/#search=hsts
Prise en charge de HPKP ou PKP (*HTTP Public Key Pinning*) :	http://caniuse.com/#search=hpkp
Prise en charge de la compression Brotli :	http://caniuse.com/#search=brotli
Prise en charge de HTTP/2 :	http://caniuse.com/#feat=http2
Prise en charge (de différentes versions) de CSP (*Content Security Policy*) et d'autres fonctionnalités associées :	http://caniuse.com/#search=csp
	http://caniuse.com/#feat=contentsecuritypolicy
	http://caniuse.com/#feat=contentsecuritypolicy2
	http://caniuse.com/#feat=upgradeinsecurerequests

En outre, grâce au service Web "User Agent Capabilities" [Réf. - 6], il est possible d'identifier le support des différents navigateurs et clients Web, y compris les bots tels que ceux utilisés par les moteurs de recherche (tels que Baidu, Bing, Googlebot, Yahoo !, Yandexbot, etc.) ou les outils et bibliothèques SSL/TLS (Java, OpenSSL, Tor, etc.) pour les capacités de sécurité fondamentales associées à HTTPS : version 1.2 de TLS (voir section "3.4.1. Versions des protocoles SSL et TLS"), SNI (voir section "3.4.2. SNI (*Server Name Indication*)", le *Forward Secrecy* (voir section "3.5.1. *Forward Secrecy*", *OCSP Stapling* (voir section "3.6.3. *OCSP Stapling*") et les tickets de session (voir section "3.7.2.2. *TLS session resumption* ou *TLS session caching*, et *TLS session tickets*").

Afin d'évaluer l'impact potentiel qu'une modification des mécanismes de sécurité HTTPS du serveur pourrait entraîner sur les clients Web, il est recommandé d'analyser les journaux (*logs*) du serveur Web et d'identifier, sur la base de l'"User Agent", le pourcentage de clients Web —d'un type donné— qui utilisent le service, afin de pouvoir évaluer et quantifier l'impact négatif que cette éventuelle modification pourrait produire.

4. Décalogue de recommandations

Décalogue de sécurité pour la mise en œuvre de HTTPS

1. Configurez l'environnement, le serveur ou l'application Web pour effectuer une redirection automatique 301 de HTTP à HTTPS pour toute requête Web.

   Passez en revue tout le contenu et le code associés à l'environnement, à l'application et aux pages Web (statiques et dynamiques) et remplacez toutes les références à "http://" par "https://" (ou par des références relatives ou absolues sans spécifier le protocole), en évitant l'utilisation de contenu mixte.

2. Utilisez HSTS (*HTTP Strict Transport Security*) pour déclarer que le serveur Web est uniquement accessible via HTTPS.

3. Utilisez la politique de sécurité du contenu (*Content Security Policy*, CSP) pour empêcher l'utilisation de contenu mixte, en indiquant que, par défaut, toutes les ressources doivent être récupérées via HTTPS. Élargir la méthode CSP en utilisant les directives "upgrade-insecure-requests" ou "block-all-mixed-content".

4. Utilisez uniquement le protocole TLS, et de préférence les versions 1.1 et 1.2 de TLS (la version 1.3 sera bientôt disponible).

5. Utilisez des clés cryptographiques ECDSA de 256 bits et/ou RSA de 2.048 bits pour les certificats numériques (X.509).

   Évaluez et sélectionnez soigneusement le type de certificat numérique à utiliser, signé à l'aide de SHA-256, l'autorité de certification (AC) qui le délivrera, la période de renouvellement, configurez correctement la chaîne de certification complète, sélectionnez l'entité (ou les entités) représentée(s) par le certificat, et garantissez sa validité à tout moment.

6. Utilisez des algorithmes d'échange de clés qui garantissent le *Forward Secrecy*, de préférence ECDHE (256 bits) ou DHE (2 048 bits), plutôt que RSA.

   Configurer le serveur Web en privilégiant les suites cryptographiques les plus robustes, classées par ordre de préférence selon le niveau de sécurité qu'elles offrent et commençant par "ECDHE-ECDSA-...", "ECDHE-RSA-..." ou "DHE-RSA-...".

7. Utilisez des algorithmes de chiffrement qui fournissent un chiffrement authentifié par l'AEAD, comme AES en mode GCM ou ChaCha20 avec Poly1305. Il convient d'utiliser des algorithmes de chiffrement symétrique offrant une sécurité d'au moins 128 bits. Il convient d'utiliser des algorithmes de *hachage* basés sur SHA-256 (ou supérieur).

8. Les certificats numériques doivent inclure des mécanismes de révocation tels que les CRL ou OCSP ou, de préférence, l'*OCSP Stapling*.

   Utilisez une AC permettant d'ajouter la directive *OCSP Must-Staple* aux certificats numériques. En complément, utilisez les capacités de notification d'*OCSP Expect-Staple*.

9. Utilisez le système HPKP (*HTTP Public Key Pinning*), ou au moins ses capacités de notification, pour identifier l'utilisation de certificats numériques illégitimes.

   Utiliser les nouveaux mécanismes de sécurité pour l'émission de certificats, tels que le *Certificate Transparency* (CT), l'en-tête *Expect-CT*, les capacités des serveurs Web à fournir des réponses SCT et les enregistrements DNS CAA.

10. Disposer de la version la plus récente (qui corrige au moins les failles de sécurité connues) des bibliothèques TLS, des serveurs et des applications Web utilisés dans la mise en œuvre de HTTPS.

ANNEXE A. Exigences pour l'analyse des sites web HTTPS

Les exigences suivantes doivent être prises en compte par les responsables des environnements Web (serveurs, applications, etc.) qui mettent en œuvre le protocole HTTPS, en vue de leur analyse dans le cadre des études réalisées par le CCN-CERT pour évaluer la mise en œuvre de HTTPS et l'état actuel de l'écosystème HTTPS dans le secteur public.

L'étude se concentre uniquement et exclusivement sur l'analyse de la mise en œuvre de SSL/TLS (*Secure Sockets Layer/Transport Layer Security*) sur les services Web (HTTP et HTTPS), et ne prévoit pas l'utilisation de SSL/TLS par d'autres services tels que SMTPS, IMAPS, POP3S, les réseaux privés virtuels (*Virtual Private Networks*, VPN), etc.

Bien que cela puisse sembler évident, il faut tout d'abord s'assurer de la disponibilité de l'environnement Web cible à tout moment, en vérifiant qu'il est au moins disponible via le service de résolution de noms ou *Domain Name System* (DNS), et qu'il offre au moins un service Web HTTPS via le port TCP/443 (ce port doit toujours être ouvert). En option, il convient d'évaluer si l'environnement Web cible offre également un service Web HTTP via le port TCP/80 (qui peut être ouvert, filtré ou fermé).

L'analyse sera effectuée par le biais de connexions TCP/IP utilisant le protocole IP version 4 (IPv4). L'environnement Web cible doit donc avoir une présence et une adresse IP publique (IPv4) sur l'Internet, indépendamment de sa présence sur d'autres réseaux basés sur le protocole IP version 6 (IPv6).

L'environnement Web cible doit être identifié par son nom ou son nom d'hôte (p. ex. www.dominio.es) et ne doit pas être identifié par son adresse IP (p. ex. 10.10.10.10) car, par exemple, l'utilisation de l'en-tête HTTP de sécurité HSTS n'est pas autorisée pour les adresses IP.

Le service HTTPS ne doit pas être proposé sur un port non standard autre que TCP/443 (p. ex. TCP/444).

L'un des premiers comportements à évaluer sera l'existence de redirections automatiques des connexions établies via HTTP (TCP/80) vers HTTPS (TCP/443).

Il faudra également évaluer la validité de la chaîne de certification ou des certificats numériques (X.509) utilisés par l'environnement Web, en allant du certificat d'une AC racine jusqu'au certificat final. Il faudra évaluer, entre autres points détaillés :

- Le nom de l'entité associée au certificat numérique et sa validité par rapport à l'identifiant de l'environnement Web analysé (nom d'hôte).
- La période de validité (ou période d'expiration) du certificat numérique.
- La validité de la chaîne de certification, en allant du certificat d'une AC racine jusqu'au certificat numérique final, en passant par les différents certificats d'AC intermédiaires.

En outre, de nombreux détails techniques associés au HTTPS des environnements Web cibles seront évalués, tels que l'état de la mise en œuvre de normes comme le *HTTP Strict Transport Security* (HSTS), le *HTTP Public Key Pinning* (HPKP), l'OSCP, les différentes versions des protocoles SSL et TLS prises en charge, etc.

L'étude sera réalisée à l'aide d'outils de numérisation et d'analyse et l'évaluation sera effectuée à l'aide du service

"SSL Server Test" [Réf. - 4] de Qualys SSL Labs et, en particulier, des API d'automatisation disponibles dans ce service.

Ce service peut également être utilisé manuellement par les responsables d'environnements Web pour auto-évaluer l'état de leurs environnements, à partir de l'adresse Web "https://www.ssllabs.com/ssltest/". Il est toujours recommandé de sélectionner l'option "*Ne pas afficher les résultats sur les tableaux*" avant de commencer l'analyse afin que l'environnement Web ne soit pas répertorié publiquement par ce service.

En résumé, les organisations souhaitant participer aux études menées par le CCN-CERT pour évaluer la mise en œuvre de HTTPS et l'état actuel de l'écosystème HTTPS dans le secteur public doivent identifier les environnements Web visés par l'analyse en fournissant le domaine principal (p. ex. cni.es) et en identifiant les serveurs et les applications Web (en utilisant le format suivant) :

- Nom du serveur Web (p. ex. www.ccn-cert.cni.es), sans inclure le protocole (p.ex. http:// ou https://) ou la ressource spécifique (p.ex. www.ccn-cert.cni.es/cursos/).

Afin de compiler la liste complète des environnements Web cibles à analyser au cours de l'étude, nous avons mis à votre disposition un document[140] à compléter par le responsable de l'Organisme. Ce document contient un onglet séparé pour chaque domaine principal à inclure dans l'analyse et, dans le même onglet, tous les serveurs Web de ce domaine qui feront partie de l'étude.

De plus, outre l'analyse HTTPS d'un serveur ou d'un site Web spécifique (p.ex. https://sede.ministerio.es), il est recommandé d'étendre l'utilisation de HTTPS à l'ensemble du domaine, c'est-à-dire au serveur Web par défaut ("www"), au domaine lui-même et à tous ses sous-domaines (p.ex. https://ministerio.es, https://www.ministerio.es et/ou https://<sous-domaine>. ministerio.es).

L'étude de la mise en œuvre de HTTPS et de l'état actuel de l'écosystème HTTPS dans le secteur public se concentrera à la fois sur l'évaluation de l'état général de la mise en œuvre de HTTPS dans le domaine principal (y compris le serveur Web par défaut, "www" et ses sous-domaines), et sur les environnements et serveurs Web spécifiques fournis par les responsables.

[140] P.ex. la feuille de calcul Microsoft Excel permettant de collecter les données de l'Organisme, de ses responsables et des serveurs Web cibles : "CCN-CERT_TargetWebEnvironments_HTTPS_2017_v1.0.xlsx".

ANNEXE B. Suites cryptographiques recommandées

La liste suivante fournit un ensemble de suites cryptographiques recommandées initialement pour configurer l'implémentation HTTPS, pour les clés RSA et ECDSA, au format OpenSSL, classées par ordre de préférence en fonction de leur niveau de sécurité et d'interopérabilité [Réf. - 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

REMARQUES :

Dans la liste ci-dessus, d'après une vision conservatrice, AES128 et CHACHA20 sont prioritaires sur AES256, en assumant que certains clients Web ne supportent pas AES-NI[74] (bien que la plupart des clients modernes le fassent). Sinon, l'AES256 (en italique) pourrait être prioritaire par rapport aux autres options.

Les clés ECDSA sont également prioritaires sur les clés RSA dans tous les cas, mais on pourrait aussi alterner la même suite pour ECDSA et, ensuite, pour RSA (en leur donnant la priorité sur toutes les autres suites basées sur ECDSA).

Sinon, l'outil "Mozilla SSL Configuration Generator"[142] peut être utilisé pour générer la configuration HTTPS initiale de différents serveurs Web, et peut spécifier le niveau de sécurité souhaité (ou le niveau de compatibilité requis avec les navigateurs et clients Web modernes ou anciens[71]) et d'autres caractéristiques associées, notamment la version du serveur Web et d'OpenSSL. Une fois la configuration générée, il est recommandé de la personnaliser et de la paramétrer en fonction des préférences et des exigences de l'environnement 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/

ANNEXE C. Références

[Réf. – 1]	**"Rapports des bonnes pratiques (BP)". CCN-CERT.** RAPPORTS 2017 https://www.ccn-cert.cni.es/informes/informes-ccn-cert-buenas-practicas-bp.html
[Réf. – 2]	**"HTTPS : TLS, pas SSL". "Vous avez eu le temps de le sécuriser : je vous en supplie, ne le lisez pas seul". Raúl Siles (DinoSec). X Journées STIC CCN-CERT.** PRÉSENTATION Décembre 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
[Réf. – 3]	**"SSL Client Test". Qualys SSL Labs.** SERVICE WEB https://www.ssllabs.com/ssltest/viewMyClient.html
[Réf. – 4]	**"SSL Server Test". Qualys SSL Labs.** SERVICE WEB https://www.ssllabs.com/ssltest/
[Réf. – 5]	**"SSL Cipher Suite Details of Your Browser". DC Sec.** WEB https://cc.dcsec.uni-hannover.de
[Réf. – 6]	**"User Agent Capabilities". Qualys SSL Labs.** WEB https://www.ssllabs.com/ssltest/clients.html
[Réf. – 7]	**"Can I Use...?"** SERVICE WEB http://caniuse.com
[Réf. – 8]	**"Transport Layer Security (TLS) Extensions: Extension Definitions" (SNI). RFC 6066. IETF.** WEB Janvier 2011 https://tools.ietf.org/html/rfc6066
[Réf. – 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)
[Réf. – 10]	**"Bulletproof SSL and TLS". Ivan Ristic. Feisty Duck.** LIVRE https://www.feistyduck.com/books/bulletproof-ssl-and-tls/
[Réf. – 11]	**"Let's Encrypt".** WEB https://letsencrypt.org
[Réf. – 12]	**"Announcing the first SHA1 collision". Google Security Blog.** ARTICLE DE BLOG Février 2017 https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html https://shattered.io
[Réf. – 13]	**"Certificate Transparency (CT)". Google.** WEB http://www.certificate-transparency.org
[Réf. – 14]	**"Certificate Transparency". RFC 6962. IETF.** WEB Juin 2013 https://tools.ietf.org/html/rfc6962

[Réf. – 15]	**"Certificate Transparency". Google.** SERVICE WEB https://www.google.com/transparencyreport/https/ct/
[Réf. – 16]	**"Expect-CT Extension for HTTP". IETF.** WEB Décembre 2016 (Projet 01 : projet-stark-expect-ct-01) https://datatracker.ietf.org/doc/draft-stark-expect-ct https://tools.ietf.org/html/draft-stark-expect-ct-01
[Réf. – 17]	**"DNS Certification Authority Authorization (CAA) Resource Record". RFC 6844. IETF.** WEB Janvier 2013 https://tools.ietf.org/html/rfc6844
[Réf. – 18]	**"CAA Mandated by CA/Browser Forum". Ivan Ristic. Qualys.** ARTICLE DE BLOG Mars 2017 https://blog.qualys.com/ssllabs/2017/03/13/caa-mandated-by-cabrowser-forum
[Réf. – 19]	**"Public Key Pinning Extension for HTTP". RFC 7469. IETF.** WEB Avril 2015 https://tools.ietf.org/html/rfc7469
[Réf. – 20]	**"BadSSL".** SERVICE WEB https://badssl.com https://github.com/chromium/badssl.com
[Réf. – 21]	**"SSL Server Rating Guide (SSL Server Test)". Qualys SSL Labs."** WEB 19 janvier 2017 (version 2009n) https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide
[Réf. – 22]	**"The Transport Layer Security (TLS) Protocol Version 1.3". RFC nnnn. IETF.** WEB Mars 2017 (Draft 19: draft-ietf-tls-tls13-19) https://tools.ietf.org/html/draft-ietf-tls-tls13-19
[Réf. – 23]	**"Transport Layer Security (TLS) Parameters" (TLS Cipher Suite Registry). IANA.** WEB https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml
[Réf. – 24]	**"HTTPS as a *ranking* signal". Google.** WEB Août 2014 https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html
[Réf. – 25]	**"Brotli Compression". Scott Helme.** WEB September 2016 https://scotthelme.co.uk/brotli-compression/
[Réf. – 26]	**"Resource Hints". W3C.** WEB Mars 2017 https://www.w3.org/TR/resource-hints/#preconnect
[Réf. – 27]	**"HTTP Strict Transport Security (HSTS)". RFC 6797. IETF.** WEB Novembre 2012 https://tools.ietf.org/html/rfc6797
[Réf. – 28]	**"HSTS Preload List". Google.** SERVICE WEB https://hstspreload.org https://opensource.google.com/projects/hstspreload
[Réf. – 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
[Réf. – 30]	**"HSTS Preload Pending List". Google.** WEB https://hstspreload.org/api/v2/pending
[Réf. – 31]	**"Mixed Content (Block All Mixed Content)". W3C.** WEB Août 2016 https://www.w3.org/TR/mixed-content/
[Réf. – 32]	**"Content Security Policy Reference".** WEB https://content-security-policy.com/
[Réf. – 33]	**"Content Security Policy 1.0". W3C.** WEB Février 2015 https://www.w3.org/TR/CSP1/ https://www.w3.org/TR/2012/CR-CSP-20121115/ (novembre 2012)

[Réf. – 34]	**"Content Security Policy Level 2 (2.0)". W3C.** WEB Décembre 2016 https://www.w3.org/TR/CSP2/
[Réf. – 35]	**"Content Security Policy Level 3 (3.0)". W3C.** WEB Septembre 2016 (projet) https://www.w3.org/TR/CSP3/ https://www.w3.org/TR/CSP/ https://w3c.github.io/webappsec-csp/ (document de travail)
[Réf. – 36]	**"Upgrade Insecure Requests". W3C.** WEB Octobre 2015 https://www.w3.org/TR/upgrade-insecure-requests/
[Réf. – 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/pRéf.er-secure-origins-for-powerful-new-features
[Réf. - 38]	**"Hypertext Transfer Protocol (HTTP/1.1): *Caching*". RFC 7234. IETF.** WEB Juin 2014 https://tools.ietf.org/html/rfc7234
[Réf. – 39]	**"SSL/TLS and PKI History". Ivan Ristic. Feisty Duck.** WEB https://www.feistyduck.com/ssl-tls-and-pki-history/
[Réf. – 40]	**"OpenSSL Cookbook". Ivan Ristic. Feisty Duck.** LIVRE https://www.feistyduck.com/books/openssl-cookbook/

centro criptológico nacional

centro criptológico nacional

www.ccn.cni.es

www.ccn-cert.cni.es

https://oc.ccn.cni.es/


ccn-cert
centro criptológico nacional
CCN
centro criptológico nacional


PUBLICACIONES
Pd
DE DEFENSA


GOBIERNO
DE ESPAÑA
MINISTERIO
DE DEFENSA
SUBSECRETARÍA DE DEFENSA
SECRETARÍA GENERAL TÉCNICA
SUBDIRECCIÓN GENERAL
DE PUBLICACIONES
Y PATRIMONIO CULTURAL