ANALYSES DES CONFIGURATIONS SSL/TLS DE SERVEURS SMTP

Arthur PROVOST – arthur.provost@insa-rennes.fr
Étudiant à l'INSA de Rennes
Olivier LEVILLAIN – olivier.levillain@ssi.gouv.fr
ANSSI

mots-clés : SSL / TLS / SMTP / SUITES CRYPTOGRAPHIQUES / VULNÉRABILITÉS

La messagerie électronique est une des applications les plus utilisées d'Internet. Il est donc naturel de s'intéresser à la sécurité des protocoles servant à l'acheminement des courriers électroniques. En première approche, il existe deux éléments à regarder : la sécurité des communications, qui est généralement assurée par SSL/TLS, et la protection des contenus, qui peut être assurée par S/MIME ou OpenPGP. C'est sur le premier point que porte cet article.

Lorsqu'on considère la messagerie électronique, il existe en pratique plusieurs acteurs et plusieurs protocoles (voir figure 1). S'il semble relativement facile de sécuriser certaines relations comme celles entre un client final et son fournisseur de service (Submission, IMAP, POP, Webmail via HTTP), la question est plus complexe pour l'acheminement proprement dit. C'est pourquoi nous nous sommes attachés à étudier les serveurs SMTP dans la nature (en particulier l'étape 2 du schéma). Cet article présente dans un premier temps le protocole SSL/TLS à travers son fonctionnement et ses attentes en termes de sécurité, puis dans un second temps les analyses effectuées sur des serveurs SMTP et leurs résultats.

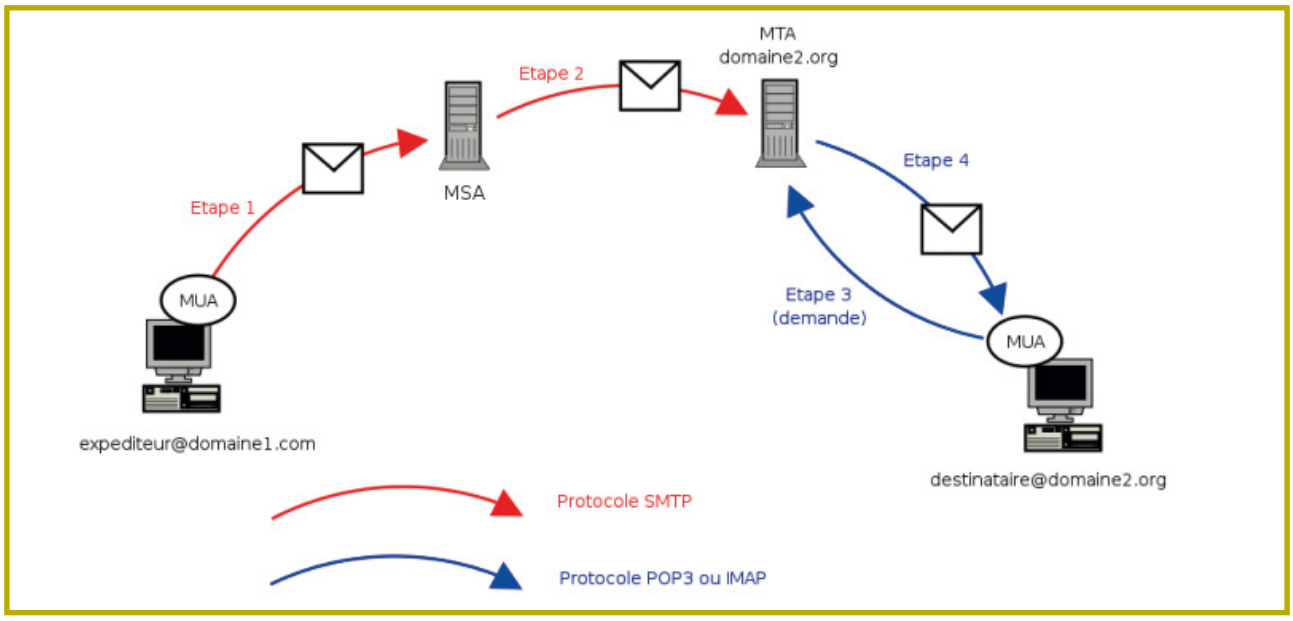
1 Rappels sur SSL/TLS

1.1 Histoire du protocole

SSL et TLS sont deux variantes du même protocole. Celui-ci vise à protéger les communications en confidentialité et en intégrité, avec une authentification unilatérale, ou, plus rarement, bilatérale. Dans la terminologie TLS, les deux parties sont appelées le client et le serveur. La première version publique du protocole, SSLv2, contenait de graves défauts dans sa conception et souffre de nombreuses attaques. SSLv3 est également vulnérable à un certain nombre d'attaques cryptographiques. Ces versions du protocole sont maintenant obsolètes et ne devraient plus être utilisées. TLS 1.0 a été l'occasion d'un petit nettoyage du protocole et de l'introduction du concept de *Perfect Forward Secrecy* (PFS) en s'appuyant sur Diffie-Hellman. Publiée en 2008, TLS 1.2 est actuellement la version du protocole la plus largement utilisée. TLS 1.3, encore en cours de standardisation, porte de nouvelles ambitions : améliorer la rapidité des communications, et leur sécurité : des algorithmes de chiffrement tels que 3DES, AES + CBC ou encore RC4 ont été retirés.

1.2 Schéma d'une connexion type

Afin d'établir une connexion sécurisée, il faut convenir des algorithmes et des clés à utiliser. Une étape de négociation (*Handshake* en anglais) est donc nécessaire. Cette étape permettra aux deux parties de transmettre des informations leur permettant de calculer les clés de session, à travers un canal de communication non sécurisé. Elle est décrite à la figure 2. Une fois cette première phase terminée, une seconde étape entre en jeu : l'utilisation des clés de session pour sécuriser et authentifier les messages échangés. Voici le déroulement général de la première étape :



Source : https://commons.wikimedia.org/wiki/File:Etapes_envoi_email.png.

Fig. 1 : Acheminement classique d'un courrier électronique.

Demande du client : la négociation commence avec l'envoi d'un message `ClientHello` au serveur. Le client propose un ensemble de suites cryptographiques qu'il supporte. Ces suites définissent les mécanismes cryptographiques utilisés pour assurer l'authentification du serveur, le chiffrement des données et leur intégrité. Ce message contient aussi la version de TLS que le client veut utiliser, ainsi que de possibles extensions.

Réponse du serveur : le serveur répond avec un message `ServerHello` s'il souhaite continuer la négociation. Sinon, il rejette la tentative de connexion avec un message `Alert`. Le `ServerHello` contient la version de TLS, la suite cryptographique et les extensions négociées.

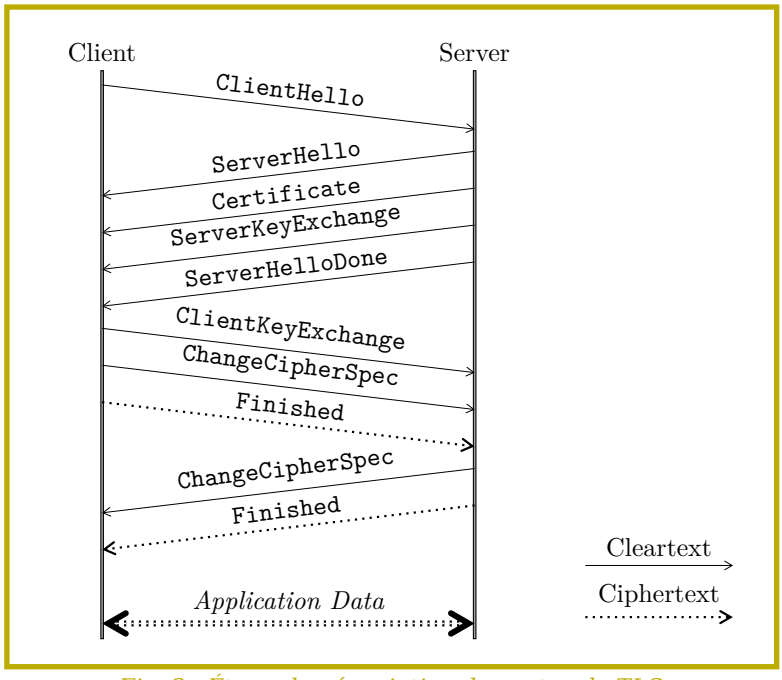


Fig. 2 : Étape de négociation du protocole TLS.

Pour choisir la suite cryptographique parmi les suites proposées par le client, il existe deux comportements classiques : le serveur peut être courtois ou directif (voir section 2.4 pour plus de détails).

Fin de la négociation : le serveur envoie son certificat, et un message `ServerKeyExchange` contenant la partie serveur du matériel cryptographique. Le client répond avec son propre matériel cryptographique (`ClientKeyExchange`) et les deux parties peuvent alors calculer un secret partagé, qui donnera des clés de session. Les messages `ChangeCipherSpec` signalent le début de la communication protégée, et les messages `Finished` garantissent a posteriori l'intégrité de la négociation, en permettant aux deux parties d'être sûres qu'ils ont calculé les mêmes clés.

1.3 Description d'une suite cryptographique

Une suite cryptographique contient un algorithme d'échange de clés, un algorithme d'authentification du tiers, un algorithme de chiffrement des données et une fonction de hachage. OpenSSL **[I]** est une boîte à outils de chiffrement contenant des algorithmes cryptographiques et une implémentation du protocole SSL/TLS en lignes de commandes.

Voici une suite disponible avec sa version 1.0.2g (2016) : `TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384`

- `ECDHE` : algorithme d'échange des clés ;
- `RSA` : algorithme d'authentification ;

- **AES256** : algorithme de chiffrement avec une clé de 256 bits ;
- **GCM** : mode de chiffrement ;
- **SHA384** : fonction de hachage avec une empreinte de 384 bits.

La partie gauche de la suite cryptographique assure l'échange des clés de sessions et l'authentification, alors que la partie droite assure la confidentialité et l'intégrité des données.

1.4 Attendus en termes de sécurité

Confidentialité et intégrité : TLS offre ces garanties sur les données échangées en permettant aux deux parties d'établir des clés de session qui resteront confidentielles, et ce même avec un attaquant de type Man-in-the-Middle (MITM). TLS propose des algorithmes de chiffrement solides (tels que AES256 ou CAMELLIA256...), ainsi que des fonctions de hachage actuellement sûres telles que SHA256 ou SHA384. Pour garantir la confidentialité et l'intégrité des échanges, les serveurs doivent donc choisir des suites cryptographiques contenant ces éléments cryptographiques.

Perfect Forward Secrecy (PFS) : c'est une propriété cryptographique qui garantit que la découverte par un attaquant de la clé privée du serveur ne compromet pas la confidentialité des communications passées entre un client et ce même serveur. L'utilisation d'algorithmes d'échange de clés reposant sur Diffie-Hellman (DHE ou ECDHE **[2]**) garantit cette propriété (à condition que les groupes utilisés soient suffisamment grands). Néanmoins, un attaquant possédant la clé privée peut mener des attaques actives contre les communications futures.

Authentification du serveur : pour éviter une attaque par usurpation, le serveur utilise généralement un certificat X.509 et sa clé privée associée pour s'authentifier auprès du client. Il est aussi possible de demander au client de s'authentifier de la même manière.

1.5 Rappels sur certaines vulnérabilités de SSL/TLS

Dans notre étude des configurations SSL/TLS des serveurs SMTP, une partie des résultats concerne l'analyse des vulnérabilités présentes. Il est de ce fait important de connaître les vulnérabilités qui vont être testées par la suite **[3, 4]**. Celles-ci peuvent être dues à des erreurs d'implémentation, des faiblesses du protocole, ou encore des faiblesses cryptographiques. Dans tous ces cas, les conséquences peuvent être désastreuses avec la divulgation d'identifiants, de coordonnées bancaires, ou encore l'établissement de communications prétendues sécurisées qui peuvent être déchiffrées par un attaquant.

1.5.1 Erreurs d'implémentation

Heartbleed (CVE-2014-0160) : c'est sûrement l'une des attaques les plus connues de ces dernières années. Elle est facilement réalisable et a été découverte après deux ans de présence dans OpenSSL. Elle nécessite l'utilisation de l'extension *heartbeat*. Ce message *heartbeat* envoyé par le client dans ses requêtes, contient de l'information (données + tailles des données) auquel le serveur répond avec un *heartbeat* de même taille que celui reçu. À cause d'un débordement de tampon en lecture, il est possible de forger une requête à laquelle un serveur vulnérable répondra en divulguant une partie de sa mémoire.

Freak (CVE-2015-0204) : une des causes de cette attaque est la législation américaine, qui imposait avant les années 2000 une restriction sur l'export de matériels cryptographiques. Freak cible les serveurs acceptant les suites cryptographiques *RSA_EXPORT*. En effet, ces suites imposent l'utilisation de clés RSA faibles (512 bits), et facilement cassables. L'attaque consiste à forcer un client vulnérable à accepter une telle suite à l'aide d'une attaque de type MITM.

Early CSS (CVE-2014-0224) : en injectant prématurément un message **ChangeCipherSpec** lors de la phase de négociation. Cette attaque permet de fixer les clés de session à une valeur connue de l'attaquant. Celui-ci peut alors déchiffrer et/ou modifier les données échangées entre le client et le serveur.

1.5.2 Faiblesses du protocole

Renégociation (CVE-2009-3555) : la renégociation TLS permet de réviser les paramètres d'une connexion. Les deux cas d'usage classiques sont le rafraîchissement des clés et l'authentification tardive du client. Il existe une vulnérabilité sur la renégociation dans laquelle le client pense qu'il exécute la première négociation avec un serveur qui pense lui exécuter une nouvelle négociation. L'attaquant peut alors injecter des données avant que le client parle au serveur. Celui-ci ne distinguera pas les données transmises par l'attaquant et celles du client légitime. Il existe deux possibilités pour corriger cette vulnérabilité : l'extension *renegociationInfo*, ou l'ajout de la suite **TLS_EMPTY_RENEGOTIATION_INFO_SCSV** dans le **ClientHello**. Il n'est pas recommandé d'utiliser ces deux mécanismes en même temps.

Logjam (CVE-2015-4000) : comme Freak, cette attaque **[5]** exploite le support des antiques suites export. Ici, le client et le serveur procèdent à un échange Diffie-Hellman dans des groupes d'une taille de 512 bits maximum. Un attaquant MITM peut alors faire croire aux deux parties que l'autre désire passer en mode Export, ce qui affaiblit alors la sécurité de la connexion. Cela nécessite néanmoins de calculer en temps réel un logarithme discret dans un groupe de 512 bits, ce qui n'est en pratique réalisable qu'avec des moyens conséquents, et après une étape de précalcul.

1.5.3 Faiblesses cryptographiques

Poodle (CVE-2014-3566) : Poodle est une attaque ciblant les systèmes autorisant SSLv3 avec une suite cryptographique utilisant le mode de chiffrement *CBC* (*Cipher Block Chaining*). Elle permet à un attaquant MITM de retrouver certains blocs du message clair. Cette attaque est souvent combinée avec une attaque de négociation à la baisse afin de forcer l'utilisation de SSLv3 entre deux parties supportant des versions plus récentes. Ainsi, Poodle permet de déchiffrer, octet par octet, des informations censées être protégées par le protocole. Cette dégradation peut être contrée avec l'utilisation d'une suite cryptographique factice : *TLS_FALLBACK_SCSV*.

Drown (CVE-2016-0800) : Drown (*Decrypting RSA with Obsolete and Weakened eNcryption*) est un oracle de padding RSA qui permet de retrouver le secret de session. L'attaque ne fonctionne que sur SSLv2. Cependant, tout autre serveur partageant la même clé privée RSA avec le serveur SSLv2 en question est aussi vulnérable à Drown, et ce, même si SSLv2 est désactivé sur ce second serveur. Le scénario est le suivant : l'attaquant collecte passivement les échanges de clés *RSA* avec le second serveur, puis exploite une faille dans SSLv2. Drown permet ainsi à l'attaquant de déchiffrer des communications reposant sur des versions récentes de TLS à l'aide d'un serveur tiers utilisant SSLv2. Il est important de noter que les versions d'OpenSSL depuis 1.0.2g ont désactivé SSLv2.

Note

Beast et les attaques par compression ne sont pas traitées dans cet article. En effet, ces attaques dépendent très fortement du modèle d'attaquant web pour lequel elles ont été conçues, et semblent compliquées à mettre en place avec SMTP.

2 Méthodologie de mesure

2.1 Liste des serveurs considérée

Les analyses de configurations TLS ont été réalisées sur la liste de domaines publiée par Cisco fin 2016. Il s'agit d'une liste gratuite du top 1 million des domaines les plus populaires, fondée sur le réseau Umbrella qui porte près de 100 milliards de requêtes par jour. Pour information, les noms de domaines de cette liste ont dans 53 % des cas une origine commerciale. La France avec ses domaines en .fr, n'est représentée que par 6 700 noms de domaines. Des précalculs ont été effectués sur cette liste à travers des requêtes DNS sur chaque nom de domaine afin de connaître les adresses IP de ces serveurs de messagerie potentiels.

```
$ dig +short domainname MX
$ dig +short name
```

Les connexions TLS ont été initiées à l'aide d'OpenSSL 1.0.2g **[6]** de façon parallélisée avec GnuParallel, et capturées avec Tshark. Voici comment lancer une connexion TLS vers un serveur mail :

```
$ openssl s_client -connect -starttls smtp ipaddress:25 -servername name
```

2.2 Compromis entre efficacité, courtoisie et aspects éthiques

Travailler sur une telle liste requiert une parallélisation des connexions pour étudier ces serveurs en un temps raisonnable. De plus, l'une des expériences présentées dans la section 2.4 initie pour chaque serveur une requête par suite cryptographique disponible dans OpenSSL afin de déterminer les suites autorisées par les serveurs, et ce, même si elles sont obsolètes. Comment obtenir une efficacité acceptable des tests en évitant l'aspect peu courtois qui se dégage des habitudes de mesures ?

Il faut d'abord mélanger la liste afin d'éviter de créer une charge importante pour les serveurs pendant l'expérience. En effet, cela permet d'espacer les connexions concurrentes sur des serveurs appartenant à un même nom de domaine.

Par ailleurs, une demande pour une utilisation particulière du réseau Renater a été effectuée, afin de rester courtois non seulement avec les serveurs ciblés, mais aussi avec le fournisseur du réseau.

Un serveur web a été installé sur la VM qui initiait les connexions, pour présenter une page web expliquant le contexte du projet, ses méthodes d'analyse et l'équipe encadrante. Cette page peut avoir rassuré certains administrateurs ayant détecté de multiples connexions TLS sur leurs serveurs. Ces analyses de configurations TLS font partie du projet SafeTLS initié par l'Agence Nationale de Recherche. Dans ce contexte, une collecte des informations délivrées par les messages `ServerHello`, une classification de celles-ci en fonction de leur sécurité, puis des analyses statistiques sont effectuées.

D'un point de vue éthique, nous avons tenu à ce que notre approche soit la plus passive possible, dans la mesure où nous n'envoyons aucune commande SMTP au-delà de la négociation (la commande `MAIL FROM` n'est, par exemple, jamais atteinte). De plus, les échanges TLS n'intègrent aucune donnée personnelle, nos connexions répétées n'ont pas été utilisées à des fins de spam et les vulnérabilités découvertes n'ont pas été exploitées. L'objectif de l'étude est d'apporter une mesure sur l'usage de SSL/TLS dans l'écosystème SMTP.

2.3 Description concrète de la mesure

L'approche est composée de deux étapes d'analyse de configurations TLS, puis d'une dernière de tests de vulnérabilité. Les deux premières étapes déterminent la version TLS négociée, la suite établie, les potentielles extensions et la taille des clés utilisées.

En partant de la classification d'OpenSSL (*LOW/MEDIUM/HIGH*) qui est discutable (RC4 et 3DES sont notés **MEDIUM**), nous avons établi une nouvelle classification : **VULN** pour les suites vulnérables à des attaques pratiques, **PASS** pour celles vulnérables à des attaques cryptographiques sans application pratique, et **OK** pour les suites ne subissant aucune attaque, même théorique.

Étape 1 : analyse de la suite cryptographique choisie par défaut par un serveur communiquant avec un client à jour. La liste de suites disponibles sur le client de test est accessible via :

```
$ openssl ciphers -v
```

Étape 2 : analyse des suites cryptographiques autorisées par un serveur de messagerie donné. Une connexion est initiée en spécifiant une unique suite cryptographique côté client :

```
$ openssl s_client -starttls smtp -cipher cipher ipaddress:25
-servername name
```

Cette approche permet de déterminer si les serveurs acceptent de mauvaises suites cryptographiques, et ce, même s'ils en choisissent une recommandée avec un client à jour. Dans ce cas, un scénario où un attaquant réussit à forcer l'utilisation d'une mauvaise suite est envisageable. Afin de tester le maximum de mauvaises suites cryptographiques (p.ex. celles contenant MD5), la version 1.0.2 d'OpenSSL datant de 2015 a été recompilée et utilisée en complément.

Étape 3 : test de vulnérabilités des serveurs. Cette phase utilise l'outil **testssl.sh** **[7]** qui suit les grandes lignes du guide fourni par l'OWASP et qui permet de déceler les vulnérabilités des serveurs et leur comportement.

Cette étape permet également de détecter le comportement du serveur face aux propositions du client :

- un serveur « courtois » cherchera dans la liste du client la première suite qu'il prend en charge ;
- un serveur « directif » parcourra sa propre liste de préférences et prendra la première suite qu'il trouve dans la liste du client.

3 Résultats

Les données de cette section sont calculées sur un ensemble de 490 000 serveurs SMTP environ (résultant de la liste présentée précédemment).

3.1 Simulation d'une connexion TLS par défaut

Le but de cette première approche est de savoir quels paramètres et quelle suite cryptographique sont choisis par défaut lorsqu'un client à jour initie une connexion avec un serveur. La version 1.0.2g d'OpenSSL a été utilisée et les résultats sont disponibles à la figure 3. Presque tous les serveurs ont choisi TLS 1.2 (94,13 %) ou TLS 1.1 (5,85 %). Concernant les suites négociées, la plupart des serveurs utilisent une suite recommandée par défaut (79,49 %) ou une suite classée **PASS** (20,19 %).

Ces suites correspondent à des algorithmes vieillissants, mais pas immédiatement attaquables. La proportion de suite cryptographique classée **PASS** peut s'expliquer par deux hypothèses : soit le serveur ne supporte aucune suite **OK**, soit il est directif et privilégie des suites plus faibles.

De plus, seuls 0,32 % des serveurs choisissent des suites faibles (**VULN**) alors que de bonnes suites étaient proposées. Ici aussi, cela signifie qu'ils ne pouvaient pas négocier de meilleure configuration, ou qu'ils

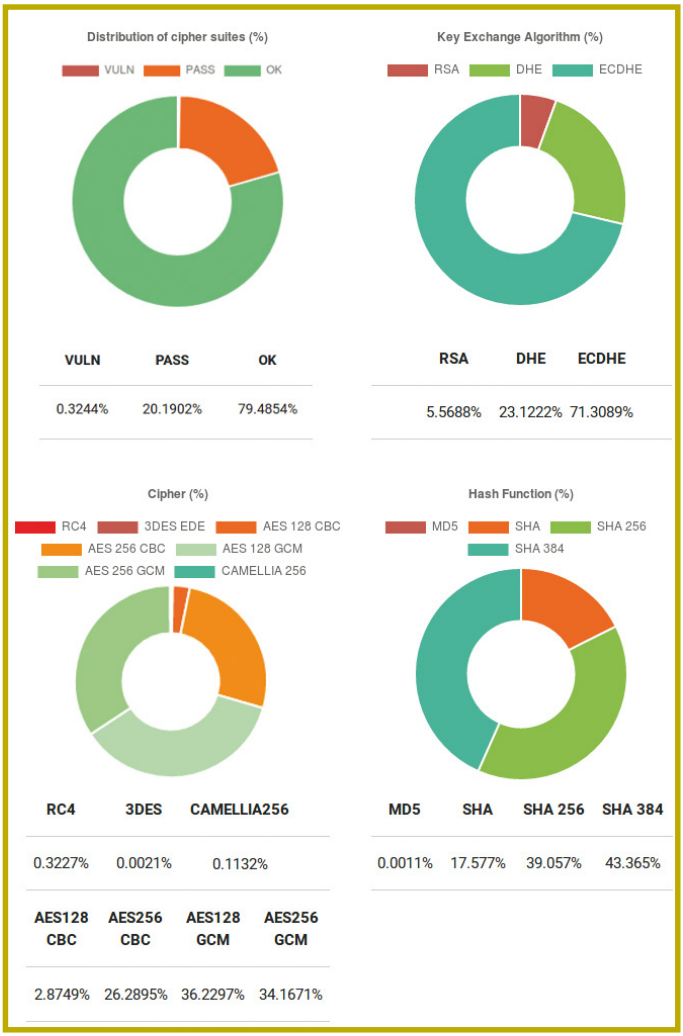


Fig. 3 : Résultat des connexions avec un client à jour.

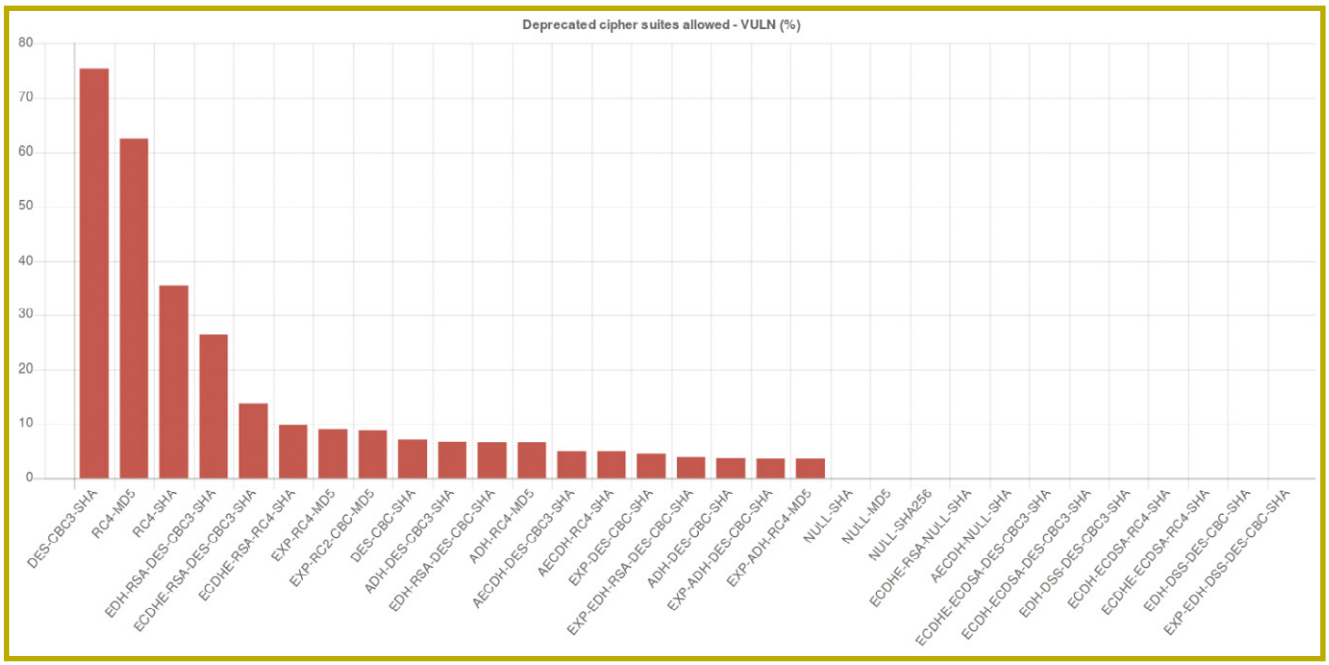


Fig. 4 : Suites cryptographiques VULN autorisées par les serveurs (toutes les suites notées ont été choisies au moins une fois par un serveur).

privilégiaient de mauvaises suites. Ces serveurs sont peu nombreux, mais sont enclins à subir des attaques concrètes (déchiffrement par attaque sur le keystream de RC4...).

En décortiquant les suites choisies, on peut voir une forte utilisation de protocoles d'échanges de clés utilisant Diffie-Hellman (94,42 %) avec DHE ou ECDHE, assurant donc la PFS.

Concernant le chiffrement, AES est très largement utilisé, avec le mode de chiffrement GCM dans plus de 70 % des configurations analysées. Néanmoins, 30 % des serveurs utilisent toujours le mode CBC qui souffre de nombreuses attaques dans SSL/TLS.

Ce premier test permet aussi de déterminer si des noms de domaines possédant plusieurs serveurs mails présentaient des configurations différentes. En effet, près de deux tiers des domaines possèdent plusieurs MX, avec une moyenne de trois MX par domaine. Il s'avère qu'un seul domaine testé utilisait des configurations différentes sur ses serveurs mails. Les différences étaient relativement faibles : changement de la longueur de la clé et de la version de SHA utilisée. Les résultats de cette première étude sont dans l'ensemble rassurants, avec une volonté des serveurs à utiliser les dernières versions de TLS, avec des suites cryptographiques recommandées.

3.2 Suites cryptographiques autorisées par les serveurs

Le but d'un attaquant actif est d'affaiblir la connexion entre le client et le serveur, en essayant d'exploiter les pires configurations que ces parties supportent. L'objectif de cette seconde analyse est donc de déterminer les pires suites (y compris les suites obsolètes) qui sont autorisées par les serveurs. La distribution des suites notées **VULN** est donnée à la figure 4.

De nombreuses suites de la catégorie **PASS** sont celles qui utilisent la fonction de hachage SHA1 pour le HMAC. Même si les attaques de collision sur SHA1 n'ont actuellement aucun impact sur la sécurité de HMAC **[8]**, celles-ci sont classées en **PASS** par précaution. *HMAC SHA256* ou *HMAC SHA384* doivent être préférés.

Certains serveurs utilisent un échange de clé anonyme (ADH, AECDH...). Il a longtemps été d'usage dans le monde SMTP d'accepter de telles configurations. Il existe en pratique encore des implémentations qui tolèrent des certificats expirés ou dont le nom ne correspond pas au serveur. Les certificats serveur ne peuvent donc pas être vérifiés avec autant de soin que dans le monde du Web. Pour ces raisons, et même si l'absence d'authentification est un problème en général, nous avons choisi de classer ces suites **PASS**.

Parmi les suites de la catégorie **VULN**, *DES_CBC3_SHA* est accepté par plus de 75 % des serveurs. Bien que 3DES ne soit plus à l'état de l'art, cette suite cryptographique est conservée pour des raisons de compatibilité avec les clients ne supportant pas *AES*. C'est d'ailleurs ce que Mozilla **[9]** explique dans sa liste de suites recommandées. Les suites *RC4_MD5* et *RC4_SHA* sont respectivement autorisées 64 % et 36 % du temps. Or il existe là encore des faiblesses cryptographiques potentiellement exploitables contre les algorithmes RC4 et MD5. Pour les lecteurs intéressés, il est possible d'obtenir l'ordre des suites cryptographiques pour Android 4.2.2 et Android 4.3 **[10]**.

3.3 Tests des vulnérabilités et des comportements des serveurs

3.3.1 Vulnérabilités

Une partie des résultats obtenus avec `testssl.sh` sont présentés à la figure 5. On y constate que les vulnérabilités relatives à SSLv2 (*Drown*) ou aux suites *EXPORT* (Freak ou Logjam), sont relativement peu présentes au sein des serveurs étudiés. Cela correspond a priori à l'utilisation de versions relativement récentes de bibliothèques TLS, qui ne supportent plus ces mécanismes. Il reste cependant étonnant que plus de 3 % des serveurs SMTP négocient encore des suites *EXPORT*.

Concernant Hearbleed et EarlyCCS, deux vulnérabilités spécifiques à OpenSSL publiées en 2014, elles représentent également une faible part des serveurs étudiés. Cette proportion est à mettre en regard du nombre de serveurs supportant l'extension *heartbeat* (plus de 45 %).

La situation est plus décevante pour les attaques POODLE (support du mode CBC avec SSLv3) et pour le support de RC4. En effet, ces mécanismes sont obsolètes et ne devraient plus être utilisés du tout aujourd'hui, mais ils sont présents sur près d'un quart des serveurs interrogés. Comme dit précédemment, cela témoigne d'une volonté de conserver une compatibilité avec de vieux clients, mais parfois l'utilisation de la cryptographie obsolète est pire que l'absence de cryptographie. La bonne solution est donc de mettre à jour les implémentations et les configurations.

Les résultats concernant la renégociation sont également assez inquiétants : si plus de 98 % des serveurs supportent l'extension *RenegotiationInfo* (qui permet d'effectuer une renégociation sécurisée), plus d'un tiers des serveurs acceptent également une renégociation non sécurisée…

Enfin, plus de 80 % des serveurs supportent l'extension *TLS_FALLBACK_SCSV*, qui permet de détecter les attaques forçant une négociation à la baisse.

Il faut cependant noter qu'il existe des limites intrinsèques aux tests réalisés. Citons par exemple Drown, qui consiste à exploiter un serveur supportant SSLv2, mais qui peut affecter d'autres serveurs partageant la même clé privée.

3.3.2 Comportements

Le deuxième objectif de cette analyse était de découvrir le comportement des serveurs pour choisir la suite cryptographique. Celui-ci peut être « courtois » ou « directif ». Il est généralement recommandé pour un serveur d'avoir un comportement directif, pour mieux contrôler la suite cryptographique qui sera négociée. Bien entendu, cette recommandation n'est pertinente que si le serveur utilise une liste de suites privilégiant les suites sécurisées, et si la liste ne contient aucune

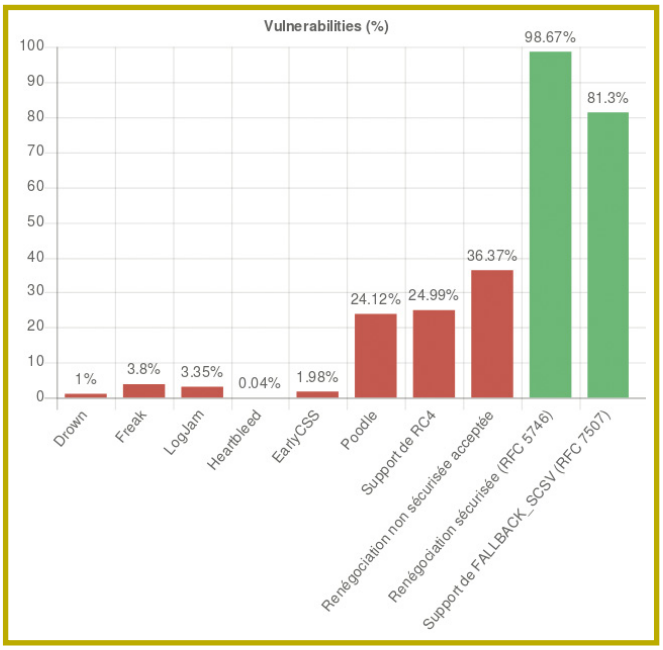


Fig. 5 : Distribution des vulnérabilités des serveurs.

mauvaise suite. Dans cette dernière expérience, plus de 56 % des serveurs ont un comportement directif, ce qui est une bonne chose s'ils respectent les conseils donnés ci-dessus.

4 Autres travaux et conclusion

Au-delà des nombreuses études réalisées sur l'écosystème HTTPS, nous avons présenté quelques résultats concernant SMTP sur SSL/TLS. Les analyses présentées se placent dans le cadre du projet SafeTLS initié en 2016. Ils font suite au travail effectué sur le protocole TLS au cours de la thèse d'un des auteurs **[11]**. Il existe également d'autres travaux sur l'état des lieux de TLS sur Internet. On peut, par exemple, citer ceux de Christopher Meyer **[12]**. Concernant l'écosystème SMTP en particulier, on peut citer des études **[13,14]** parues en 2015 et plus particulièrement les analyses proposées par Binu Ramakrishnan **[15]** ; un travail sur les certificats (vérification, profondeur de la chaîne de certificats...) a été effectué, et peut compléter le présent article. Les résultats obtenus sont comparables aux nôtres : plus de 80 % des connexions standards assurant la PFS et une large utilisation de TLS1.2. Néanmoins TLS 1.0, qui était utilisé par défaut dans 42,53 % des serveurs dans cette étude en 2015, est passé à 5,85 % en 2017.

Même s'il existe encore un certain nombre de serveurs qui ne sont pas à jour, il y a donc quelques raisons d'être optimiste quant à l'écosystème SMTP (large utilisation de TLS 1.2 et de suites cryptographiques modernes). Un des défis majeurs pour améliorer le paysage est désormais d'éviter qu'un attaquant actif puisse facilement imposer l'utilisation d'un mode dégradé en clair. Une proposition similaire au mécanisme HSTS (*HTTP Strict Transport Security*), intitulée MTA-STS est en cours de standardisation à l'IETF et devrait permettre de définir une politique plus stricte. ■

Remerciements

Nous remercions Gildas Avoine, Cristina Onete et Florian Maury pour leurs expertises, leurs conseils et leurs relectures constructives.

Références

[1] OpenSSL. *Cryptography and SSL/TLS Toolkit*, 2017.

[2] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller. RFC 4492. *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)*, 2006.

[3] miTLS. A Zoo of TLS Attacks, 2017. https://www.mitls.org/pages/attacks/

[4] Y. Sheffer, R. Holz, and A. Saint-Andre. RFC 7457. *Summarizing Known Attacks on Transport Layer Security*. 2015.

[5] Weak DH Team. *Imperfect Forward Secrecy : How Diffie-Hellman Fails in Practice*. ACM-CCS 2015.

[6] I. Ristic. *OpenSSL Cookbook : A short Guide to the Most Frequently Used OpenSSL Features and Commands*. Feisty Duck, 2015.

[7] D. Wetter. *Testssl*, 2016.

[8] ANSSI. *Recommandations de sécurité relatives à TLS*, 2016.

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

[10] OpCo. *Why Android SSL was Downgraded from AES256-SHA to RC4-MD5*, 2013.

[11] O. Levillain. *A study of the TLS ecosystem*. Thèse, 2016.

[12] C. Meyer, *Strengthening Web Authentication through TLS - Beyond TLS Client Certificates*, 2014.

[13] Wilfried Mayer, Aaron Zauner, Martin Schmiedecker, Markus Huber, *No Need for Black Chambers : Testing TLS in the E-mail Ecosystem at Large*, 2015, https://arxiv.org/abs/1510.08646.

[14] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, J. Alex Halderman - *Neither Snow Nor Rain Nor MITM... : An Empirical Analysis of Email Delivery Security*. Internet Measurement Conference 2015.

[15] B. Ramakrishnan, *Analysis of TLS in SMTP World*, 2015.