Messaging, Malware and Mobile Anti-Abuse Working Group

Implémentation de la Forward Secrecy pour la sécurisation des échanges de données Recommandations initiales du M³AAWG

Janvier 2016

Introduction

Le déploiement du chiffrement opportuniste, comme décrit dans le document "Chiffrement TLS des e-mails : Recommandations Initiales du M3AAWG" est une excellente entrée en matière pour la sécurisation des échanges d'emails entre opérateurs de messagerie électronique. Cependant, les utilisateurs de TLS ne sont pas toujours conscients que les connexions mettant en œuvre le TLS sans utiliser le « Forward Secrecy » (parfois traduit « confidentialité persistante » en français) ne sont pas suffisamment sécurisées. Si une personne mal intentionnée capture et enregistre une partie du trafic chiffré, puis parvient à se procurer les clés privées utilisées pour son chiffrement, elle sera alors en mesure de déchiffrer et lire l'intégralité du trafic capturé.

Le Forward Secrecy est un ensemble de protocoles cryptographiques qui permet de pallier cette vulnérabilité. L'activation du Forward Secrecy en conjonction avec le chiffrement TLS permet de protéger le trafic potentiellement capturé de toute possibilité de déchiffrement. Ce document vise à démontrer comment l'implémentation du Forward Secrecy[1] et de clés éphémères[2] peut protéger les données transmises au cours de ces sessions, et surtout, comment elle peut les protéger de tout déchiffrement ultérieur si une personne non fiable et ayant accès aux données venait à se procurer les clés qui ont été utilisées.

Raisons d'utiliser le Forward Secrecy

Le chiffrement/déchiffrement du contenu des messages qui a lieu au sein d'une session SSL/TLS est effectué à l'aide d'un algorithme de cryptage *symétrique* très performant, comme AES. Ces cryptages sont dits symétriques car ils font usage de la même clé aussi bien pour chiffrer que pour déchiffrer. La « clé secrète » utilisée pour ce cryptage est échangée préalablement au sein d'une communication chiffrée par un cryptage asymétrique qui assure une meilleure sécurité.

La cryptographie asymétrique utilise une paire de clés publique/privée, ce qui signifie que l'on utilise des clés différentes pour chiffrer et déchiffrer les données. La clé publique est faite pour être mise à disposition de tous, mais la clé privée doit être scrupuleusement gardée secrète. En effet, quiconque en sa possession pourra déchiffrer toutes les données chiffrées avec la clé publique associée. Avec SSL/TLS, la paire de clés publique/privée est généralement une paire de clés RSA créée à l'occasion d'une demande de certificat signé. Une fois générées, ces paires de clés publique/privée sont très rarement voire jamais changées par leurs propriétaires.

[1] Confidentialité persistante, https://fr.wikipedia.org/wiki/Confidentialit%C3%A9_persistante
[2] Ephemeral key (en anglais), https://en.wikipedia.org/wiki/Ephemeral_key

Cette approche fonctionne en général plutôt bien à l'exception d'un scénario : Qu'en est-il si une personne mal intentionnée capture tout ou partie d'une communication chiffrée en SSL/TLS et parvient *également* à obtenir une copie de la clé privée de l'opérateur ? Si l'opérateur n'utilisait pas une suite cryptographique offrant le Forward Secrecy pour protéger sa clé privée, l'attaquant aurait alors, en sa possession, tous les moyens nécessaires pour déchiffrer a posteriori *l'intégralité* du trafic capturé associé à cette clé privée.

Un attaquant est-il réellement en mesure de capturer du trafic ? Oui ! Certains attaquants peuvent écouter (et enregistrer) l'intégralité du trafic circulant à travers les connexions qui sont à leur portée. Dans d'autres cas, le trafic peut être redirigé par des attaquants spécifiquement dans le but de pouvoir le copier.
Comment un attaquant pourrait-il se procurer la clé privée ? Principalement de deux façons :

1. La clé privée pourrait être révélée accidentellement suite à une vulnérabilité logicielle. Par exemple, les clés privées pouvaient être visibles par des attaquants à cause de la faille HeartBleed. C'est pour cette raison que les sites impactés ont dû générer de nouvelles paires de clés publique/privée et obtenir de nouveaux certificats.

2. Puisque beaucoup de sites stockent leur clé privée dans un simple fichier plutôt que dans un module matériel de sécurité (HSM, Hardware Security Module), quiconque ayant accès au fichier concerné et aux clés qu'il contient, pourrait déchiffrer tout le trafic associé. Voici quelques exemples de stratégies d'accès à ces fichiers :
    - Exploiter une faille humaine en ciblant un administrateur système ou un autre utilisateur disposant des privilèges nécessaires (corruption, chantage, menaces physiques, etc.)
    - Accéder à une copie mal sécurisée du fichier, par exemple dans le cadre d'une sauvegarde non cryptée sur un site distant, ou d'un piratage visant spécifiquement à accéder au contenu de ce fichier
    - Divulgation accidentelle ou intentionnelle d'une version actuelle ou ancienne de la clé privée par un administrateur non conscient du caractère sensible de cette clé
    - Décision juridique imposant la divulgation de la clé

Utilisation du Forward Secrecy pour la sécurisation des échanges de données

Heureusement, l'échange de clés éphémères apporte une solution à ce problème. Quand un site utilise un mécanisme d'échange de clés qui supporte le Forward Secrecy, tel que Diffie-Hellman[3] Ephémère (DHE) ou Diffie-Hellman basé sur des Courbes Elliptiques (ECDHE/EECDH), une nouvelle paire de clés publique/privée est créée pour chaque connexion et détruite immédiatement après usage. Avec cette approche, même si le trafic a été capturé et si la sécurité de la clé privée RSA a été compromise, il sera impossible à un attaquant de décrypter rétrospectivement les communications enregistrées.

En utilisant des mécanismes d'échange de clés éphémères, une attention particulière doit être apportée pour s'assurer que les paramètres Diffie-Hellman soient de longueur suffisante et, par conséquent, suffisamment robustes.
Dans certaines circonstances, les clés DH par défaut peuvent n'être que de 1024 bits. Heureusement, les versions actuelles des principales librairies cryptographiques comme OpenSSL permettent désormais l'utilisation de clés DH allant jusqu'à 4096 bits.[4]

[3] Échange de clés Diffie-Hellman, https://fr.wikipedia.org/wiki/%C3%89change_de_cl%C3%A9s_Diffie-Hellman
[4] Alex Halderman and Nadia Heninger "How is NSA breaking so much crypto?," Freedom to Tinker, October 14, 2015 (en anglais), https://freedom-to-tinker.com/blog/haldermanheninger/how-is-nsa-breaking-so-much-crypto/

Bien qu'il ne soit pas très fréquent de se baser sur des connexions sécurisées par clés RSA publique/privée pour échanger la clé secrète en toute sécurité lors de l'établissement d'une connexion, ces algorithmes ont encore un rôle majeur dans la cryptographie SSL/TLS. Ils constituent la base pour l'authentification du système distant et assurent qu'un site pirate ne tente pas de mener une attaque par « l'homme du milieu » (Man-in-the-Middle). Bien que les clés RSA de 2048 bits soient les plus courantes de nos jours, la plupart des autorités de certification peuvent tout à fait générer des CSRs avec des clés RSA de 3072 ou 4096 bits. (Note : avant de passer à des clés RSA 3072 ou 4096 bits, pensez à vérifier l'impact que l'utilisation de clés de cette taille peut avoir sur le débit et la capacité de votre système).

Conseils pour l'implémentation du Forward Secrecy

Lorsque vous configurez un serveur web ou une autre application utilisant SSL/TLS, assurez-vous de sélectionner des suites cryptographiques « éphémères ». Ces suites nécessitent l'utilisation de paramètres Diffie-Hellman adéquats. Le procédé exact pour sélectionner ces suites peut varier selon les librairies de chiffrement. Il est donc nécessaire de vous référer à la documentation de celle que vous utilisez.

Voici quelques exemples pour illustrer les prérequis génériques :

1. *Implémenter le Forward Secrecy*
   a) Exemple : OpenSSL 1.0.1p avec nginx[5] 1.9.5.
      i. Commencez par générer des paramètres Diffie-Hellman appropriés (les paramètres par défaut sont trop faibles) :
         - `cd /etc/ssl/certs`
         - `openssl dhparam`[6] `-out dhparam.pem 4096`

         Note : La génération du fichier dhparam.pem prendra un certain temps.
      ii. Puis, dans le fichier de configuration nginx.conf, ajoutez une suite cryptographique appropriée :
         - `ssl_ciphers 'AES256+EECDH:AES256+EDH';`
         - `ssl_dhparam /etc/ssl/certs/dhparam.pem;`

   b) Exemple : OpenSSL 1.0.1p avec Dovecot 2.2.
      i. Créez un certificat comme décrit ici (en anglais) : http://wiki2.dovecot.org/SSL/CertificateCreation
      ii. Généralement, dans le fichier conf.d/10-ssl.conf :
         - `ssl_protocols = !SSLv2 !SSLv3`
         - `ssl_prefer_server_ciphers = yes`
         - `ssl_cipher_list =`
           `EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA384:EECDH+ECDSA+SHA256:EECDH+aRSA+SHA384:EECDH+aRSA+SHA256:EECDH+aRSA+RC4:EECDH:EDH+aRSA:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS:!RC4`

[5] Strong SSL Security on nginx (en anglais), https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html
[6] "dhparam", Open SSL Cryptography and SSL/TLS Toolkit (en anglais), https://www.openssl.org/docs/apps/dhparam.html

c) Exemple : OpenSSL 1.0.1p avec Postfix 2.6.

i. Générez le certificat :

- `openssl dhparam -out dh_4096.pem 4096`

ii. Dans le fichier main.cf

- `smtpd_tls_eecdh_grade = strong # 128bit, "ultra" pour 196bit`
- `smtpd_tls_4096_param_file = /etc/postfix/dh_4096.pem`
- `smtpd_tls_mandatory_exclude_ciphers = aNULL, MD5, DES, ADH, RC4, PSD, SRP, 3DES, eNULL`
- `smtpd_tls_protocols= !SSLv2,!SSLv3`
- `smtpd_tls_mandatory_protocols= !SSLv2,!SSLv3`
- `tls_preempt_cipherlist = yes`
- `smtpd_tls_loglevel = 1`
- `smtp_tls_loglevel = 1`

Note : Les librairies de chiffrement et les suites cryptographiques évoluent avec le temps. Bien que les exemples ci-dessus soient considérés comme une proposition solide au moment de la rédaction de ce document, il est très probable que les prérequis de configuration soient amenés à changer et doivent être vérifiés.

2. Vérifier la configuration

Utilisez un outil de vérification SSL capable de vérifier le Forward Secrecy. Cela peut être fait par l'intermédiaire d'un site web dédié ou par un script shell.

d) Vous pouvez par exemple utiliser des sites (en anglais) comme https://www.ssllabs.com/ssltest/ ou https://ssl-tools.net/

e) Vérifiez la simulation du "handshaking" sur un de ces sites pour valider la suite de chiffrement qui doit être utilisée.

Conclusion

A la différence des autres mécanismes d'échanges de clés, les échanges de clés éphémères Diffie-Hellman créent un environnement rendant totalement impossible le déchiffrement de données capturées lorsque les clés privées ont été volées a posteriori. Le groupe de travail portant sur la messagerie, les logiciels malveillants et la lutte contre l'abus par voie mobile recommande aux acteurs du secteur de mettre en œuvre la confidentialité persistante en conjonction avec TLS afin d'améliorer la confidentialité globale des échanges.[7] L'adoption généralisée de ces pratiques bénéficiera à toutes les parties prenantes.

Ces recommandations se concentrent sur l'un des aspects cryptographiques de la sécurité de la messagerie. Nous vous recommandons de prendre connaissance des autres documents publiés par le M³AAWG sur les autres sujets relatifs à la cryptographie et de garder un œil sur les documents à venir.

[7] Yan Zhu and Yan Zhu, "Why the Web Needs Perfect Forward Secrecy More Than Ever," Electronic Frontier Foundation, April 8, 2014 (en anglais), https://www.eff.org/deeplinks/2014/04/why-web-needs-perfect-forward-secrecy

Références

Vincent Bernat, SSL/TLS & Perfect Forward Secrecy, http://vincent.bernat.im/fr/blog/2011-ssl-perfect-forward-secrecy.html

OpenSSL Cryptography and SSL/TLS Toolkit, "dhparam", https://www.openssl.org/docs/apps/dhparam.html

Section "Forward secrecy" de "Transport Layer Security", https://en.wikipedia.org/wiki/Transport_Layer_Security#Forward_secrecy

Alex Halderman and Nadia Heninger "How is NSA breaking so much crypto?", Freedom to Tinker, 14 octobre 2015, https://freedom-to-tinker.com/blog/haldermanheninger/how-is-nsa-breaking-so-much-crypto/

Ivan Ristić, "Increasing DHE strength on Apache 2.4.x", Blog: Ivan Risti , 15 août 2013, http://blog.ivanristic.com/2013/08/increasing-dhe-strength-on-apache.html

"Strong SSL Security on nginx", https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html

Yan Zhu and Yan Zhu, "Why the Web Needs Perfect Forward Secrecy More Than Ever", https://www.eff.org/deeplinks/2014/04/why-web-needs-perfect-forward-secrecy

Comme pour tous les documents de bonnes pratiques que nous publions, merci de consulter le site du M³AAWG (www.m3aawg.org) pour des éventuelles mises à jour de ce document.

M3AAWG100-French