MEGA

Cryptography Report

Auteur:

NAILI Louis

louis.naili@student.junia.com

AP5 – P63 - ISEN LILLE

Date de rendu:

14 Novembre 2021

JUNIA ISEN

Abstract

Ce document se veut être un rapport couplé à une analyse technique au sujet du chiffrement adopté par le service d'hébergement de fichiers en ligne MEGA.

Table des matières

1. Introduction

1.1. Historique

MEGA est un service d'hébergement de fichiers en ligne créé par la société *MEGA Limited*.
Le nom MEGA est un acronyme récursif signifiant « MEGA Encrypted Global Access »

Le service a été créé le 19 janvier 2013 par Kim SCHMITZ, aussi connu sous le nom de Kim DOTCOM notamment pour la création du site WEB Megaupload.
Il faudra noter que la date de création du service n'est pas anodine car elle correspond à l'anniversaire de la fermeture de la plateforme Megaupload.
Cette dernière avait été fermée par les autorités américaines pour cause de mise à disposition de contenu soumis à copyright.

Beaucoup d'aspects de la plateforme MEGA semblent avoir été pensés pour éviter les affaires judiciaires associées avec Megaupload.

1.2. La sécurité selon MEGA

À une époque où la politique de sécurité d'une majeure partie des acteurs de l'informatique est la sécurité par l'obscurité, MEGA s'engage dans une démarche de transparence.
L'intégralité du code utilisé pour le chiffrement pour les différentes plateformes est par conséquent mis à disposition sur le Git de MEGA.

La politique de sécurité chez MEGA consiste à chiffrer l'intégralité des fichiers à synchroniser depuis l'appareil de l'utilisateur avant de transférer ces derniers via un canal sécurisé vers les serveurs MEGA.

Le service MEGA est accessible depuis tous type d'appareil disposant d'un navigateur WEB respectant le standard Javascript en vigueur.
Il est également possible d'utiliser un client, dit lourd, disponible pour toutes les plateformes.
Le chiffrement AES est effectué depuis le client, quelle qu'en soit sa nature, les calculs se font en Javascript.
Les fichiers sont donc chiffrés avant de partir vers les serveurs MEGA, l'hébergeur n'a donc pas de possibilité d'accéder au contenu des fichiers mis en ligne.
Une fois un fichier chiffré, il est envoyé via un canal sécurisé TLS 1.3 aux serveurs de MEGA.

Le fait de chiffrer systématiquement les fichiers mis en ligne peut également être vu comme une forme de protection contre des poursuites légales au sujet du contenu de ces derniers.
En effet, dans la mesure où MEGA n'a jamais accès aux fichiers déchiffrés, MEGA ne peut pas être tenu comme responsable du contenu de ces derniers.
Cela peut aussi être vu comme une rébellion envers les états pratiquant la surveillance de masse et demandant aux hébergeurs l'accès aux données de certains utilisateurs.

2. Fonctionnement

2.1. Création d'un compte

Lors de l'inscription, l'utilisateur doit rentrer un ensemble d'informations contenant son nom, prénom, adresse courriel et un mot de passe devant excéder une longueur de 8 caractères.

Avant d'être transmis, le mot de passe sera quant à lui haché en utilisant PBKDF2 sur 100 000 itérations.
Il serait préférable ici d'utiliser Argon2 ou Scrypt, cependant ces derniers ne sont pas implémentés en Javascript au travers de l'API Web_Crypto et sont par nature plus gourmands en mémoire vive, ce qui poserait un problème pour les systèmes les moins performants.

Une clef de 128bits nommée ici MasterKey est générée à partir du générateur de nombres pseudo aléatoires de Web_Crypto, de la même manière, une autre valeur nommée ClientRandomValue est générée.

La recommandation pour stocker les mots de passes en base de données est de le hacher et de le saler avec l'identifiant de l'utilisateur.
Chez *MEGA*, le fonctionnement est différent car le chiffrement serait dépendant de l'adresse courriel utilisée, ce qui signifierait que la clef serait caduque lors d'un changement d'adresse courriel.

Le sel est calculé à partir d'un hash SHA256 comme suit :
Salt = SHA-256("mega.nz" || Padding || ClientRandomValue)
Avec Padding correspondant au caractère 'P' répété 93 fois, de manière à ce que la concaténation de « mega.nz » et Padding fasse 200 caractères.
L'utilité du Bourrage étant de s'assurer que l'on ne puisse pas faire d'attaque par timing.

Une fois le sel calculé, le navigateur va calculer la clef dérivée, nommée ici DerivedKey, de la manière suivante :
DerivedKey = PBKDF2-HMAC-SHA-512(Password , Salt , Iterations , Length)
Password étant le mot de passe de l'utilisateur, Salt le sel calculé précédemment, Iterations=100000 et Length défini à 256 bits.
C'est ici que la fonction de hachage PBKDF2 est appelée.

Les 128 premiers bits de DerivedKey en partant du msb représentent DerivedEncryptionKey.
La clef MasterKey créée précédemment est ensuite chiffrée en AES avec cette dernière :
EncryptedMasterKey = AES-ECB(DerivedEncryptionKey , MasterKey)

Le hash SHA256 des 128 bits restants de DerivedKey représente DerivedAuthenticationKey, calculé comme suit :
HashedAuthenticationKey = SHA-256(DerivedAuthenticationKey)

Les données suivantes sont transmises aux serveurs MEGA :
Nom, Prénom, Adresse Courriel, ClientRandomValue, EncryptedMasterKey, HashedAuthenticationKey

Ces données seront stockées telles qu'elles sur les serveurs MEGA, tous les calculs précédents ayant été effectués par le navigateur du client, MEGA n'y a pas accès.

Le serveur envoie un lien de confirmation permettant l'activation du compte par courriel.

Une fois que l'utilisateur valide son compte, il lui est demandé de se connecter une première fois afin de générer ses clefs.

Cette étape de génération des clefs peut être assez longue. En effet, il faut générer 3 trousseaux différents depuis le navigateur de l'utilisateur.
Une paire de clef RSA 2048 pour le partage de fichiers, une paire de clefs ED25519 256bits servant essentiellement à signer les autres clefs, ainsin qu'une paire de clefs Curve25519 256bits utilisée pour le service MEGAchat.
Toutes les clefs privées sont ensuite chiffrées en AES en utilisant MasterKey.

Les clefs publiques et les clefs privées chiffrées sont transmises aux serveurs MEGA pour être stockées.

2.2. Connexion à un compte

L'utilisateur entre son adresse courriel et son mot de passe, dans le formulaire prévu à cet effet.
Une requête est envoyée au serveur MEGA contenant l'adresse courriel uniquement.

Dans le cas ou l'adresse est trouvée sur le serveur MEGA, la réponse du serveur sera le sel calculé comme pour la création du compte (voir plus haut) :
Salt = SHA-256("mega.nz" || Padding || ClientRandomValue)

Dans le cas contraire, le serveur renvoie également une valeur :
Pour éviter que des spammeurs n'utilisent l'API pour récupérer des adresses courriel valides, lors d'une tentative infructueuse une valeur nommée ServerRandomValue est calculée par les serveurs MEGA, elle fait 128 bits. Elle sera stockée en base de donnée pour ne pas vendre la mèche si un spammeur réessaie d'obtenir une réponse du serveur avec la même adresse courriel.
La valeur renvoyée est calculée comme suit :
Salt = SHA-256(Email || "mega.nz" || Padding || ClientRandomValue)
Avec Padding = 200 – (Email.length() + «mega.nz».length())
Pour améliorer la robustesse envers des attaques de timing, un délai aléatoire est introduit pour cette réponse API entre 0 et 50ms, 50ms représentant le temps maximum pour obtenir une réponse de la part de la base de données.
Ces calculs s'effectuant coté serveur, il faut également prévoir le cas d'une attaque de type déni de service. Pour cette raison, la taille maximale acceptée pour le champ courriel est de 190 caractères.

Une fois ce sel récupéré, le navigateur du client peut calculer la clef dérivée DerivedKey comme suit :
DerivedKey = PBKDF2-HMAC-SHA-512(Password , Salt , Iterations , Length)
Password étant le mot de passe de l'utilisateur, Salt le sel calculé précédemment, Iterations=100000 et Length défini à 256 bits.

Ensuite il faut calculer DerivedEncryptionKey. Pour se faire, il suffit de prendre les premiers 128 bits de DerivedKey en partant du msb.
La seconde partie de DerivedKey représente AuthenticationKey.

Le navigateur de l'utilisateur envoie au serveur MEGA l'adresse courriel ainsi que AuthenticationKey.

Dès la réception de ces informations, le serveur calcule le hash SHA256 de AuthenticationKey et le compare avec celui en base de données.
La raison pour laquelle le calcul du hash se fait coté serveur et non client est dans le but d'éviter qu'un attaquant puisse envoyer un hash directement vers le serveur et gagner l'accès aux données de l'utilisateur.
Généralement, les mots de passes sont stockés hachés en base de données, ce qui implique qu'en cas de fuite de données, le mot de passe de l'utilisateur n'est pas exposé.
Ici, on imagine qu'un attaquant a accès à la base de données en lecture. Il n'est pas en mesure de se connecter en tant qu'un utilisateur car il ne dispose que du hash.
De plus, pour éviter les attaques timing, la vérification HMAC double est effectuée, l'idée étant de hacher plusieurs fois la chaine de la manière suivante :

H(key ‖ H(key ‖ message))

Si le mot de passe de l'utilisateur n'était pas erroné, alors le serveur MEGA renverra les informations suivantes :

EncryptedMasterKey, EncryptedRsaPrivateKey, EncryptedSessionIdentifier

Le navigateur peut désormais déchiffrer MasterKey à l'aide de EncryptedMasterKey et DerivedEncryptionKey.
Il peut maintenant déchiffrer RsaPrivateKey avec MasterKey.
Enfin, SessionIdentifier peut être déchiffré avec RsaPrivateKey.
SessionIdentifier est un token aléatoirement généré par le serveur qui est chiffré par ce dernier à l'aide de RsaPublicKey avant d'être envoyé au client.
Ce token déchiffré doit être renvoyé au serveur MEGA permettra d'assurer que l'échange entre le serveur MEGA et le navigateur est sécurisé.

2.3. Mise en ligne d'un fichier

L'utilisateur ajoute un fichier à la liste des fichiers à téléverser, le navigateur se met au travail.

Une clef de 128 bits est créée à base de nombres pseudo aléatoires nommée FileKey.
Cette clef va servir à chiffrer par blocs de 128 bits le fichier via AES.
Une fois chiffré, le fichier est transmis aux serveurs MEGA.
La FileKey est, quant à elle, chiffrée en AES avec la MasterKey avant d'être stockée sur les serveurs de MEGA.

2.4. Téléchargement d'un fichier

Pour télécharger un fichier, il faut le lien vers le fichier ainsi que la FileKey.

Chaque fichier possède un identifiant unique, cet identifiant est créé coté serveur par MEGA.
Si je souhaite partager un fichier avec une personne il me suffit de cliquer sur partager le fichier, ce qui me donnera un lien de téléchargement incluant la FileKey.

Le lien sera formé comme suit :
www.mega.nz/#!Base64(FileId)!Base64(FileKey)

2.6. Récupération de mot de passe

La clef de chiffrement MasterKey est stockée chiffrée sur les serveurs de MEGA sous le nom de EncryptedMasterKey. Elle est chiffrée à l'aide d'un dérivé du mot de passe de l'utilisateur.
Cette clef MasterKey est requise pour accéder aux fichiers et n'est pas connue de MEGA. Par conséquent, si l'utilisateur oublie son mot de passe, alors il perdra l'accès à ses fichiers. Se souvenir de son mot de passe est donc critique.
MEGA n'intègre pas de mécanismes de récupération de mot de passe à base de courriel uniquement. Pour récupérer l'accès à ses données, il est nécessaire d'avoir effectué au préalable une sauvegarde de MasterKey.
Le mécanisme de récupération permet de rétablir l'accès aux fichiers suite à un changement de mot de passe si la valeur de MasterKey est connue.

2.7. Bug Bounty

Le code de MEGA est intégralement accessible depuis leur Git. Le code est open source, tous les utilisateurs le souhaitant peuvent consulter le code et en juger la qualité afin de lui faire ou ne pas lui faire confiance.
MEGA a d'ailleurs une chasse aux bugs permanente avec une récompense pouvant aller jusque 10 000€ pour les bugs les plus critiques, les modalités sont disponibles sur leur site.

MEGA utilise la librairie AsmCrypto, qui depuis la version 2.0 utilise l'API Web_Crypto, soit l'implémentation la plus standard et la mieux maintenue des fonctions cryptographiques en Javascript.
Si une faille est découverte dans l'implémentation de Web_Crypto, MEGA ne sera pas le seul concerné et la faille sera résolue plus rapidement du fait que des milliers de sites reposent sur les fonctions Web_Crypto.

3. Critiques

Le chiffrement de MEGA n'est pas infaillible de lui-même, il est du ressort de l'utilisateur de s'assurer de respecter les évidences en matière de sécurité.
Aucun service ne peut s'assurer du fait que l'appareil de l'utilisateur ne soit pas compromis en premier lieu, le chiffrement serait caduc si ce scénario venait à se produire.
La qualité du chiffrement AES étant dépendante de la source d'entropie et cette dernière provenant partiellement du mot de passe de l'utilisateur, il est du ressort de l'utilisateur de choisir un mot de passe complexe.

Lors de l'annonce de la création de MEGA, de nombreuses personnes se sont insurgées en annonçant que faire du chiffrement en Javascript était insensé.
Outre les critiques des personnes qui pensaient que MEGA se contentait de passer par un canal TLS et appelait cela un hébergement chiffré.
Les critiques visaient 3 vecteurs principaux : la qualité de l'entropie, la rapidité d'exécution, le manque d'isolation en mémoire.

3.1. Entropie

La question de l'entropie est résolue avec la qualité des générateurs de nombres pseudos aléatoires (PRNG) inclus dans les navigateurs récents, notamment l'API Web_Crypto.
Les mouvements de la souris de l'utilisateur, ses actions clavier, son adresse courriel, ainsi que son mot de passe sont utilisés pour renforcer l'entropie.

3.2. 2FA

La critique a, pendant un temps, ciblé MEGA pour son manque de support de l'authentification 2 facteurs. Ce problème est résolu depuis début 2015, le 2FA est supporté et peut être paramétré au souhait de l'utilisateur au travers de l'interface web.

3.3. Vitesse d'exécution

La rapidité d'exécution était critiquée à l'époque. Cependant, à ce jour, il est parfaitement commun de voir des sites WEB charger des dizaines de mégaoctets sur la machine du client ainsi qu'effecteur de nombreux calculs.
Reprenons les mots de Mathias ORTMANN (CTO de MEGA) "*JS is not the fastest but it is fast enough* ". En 2013, il adresse les critiques à ce sujet en annonçant une vitesse de 30Mo/s de vitesse de chiffrement en Javascript.

3.4. Sandboxing

Les données et clefs sont toujours stockées en mémoire à un moment donné par un programme.
Dans le cas de MEGA, la crainte serait qu'un utilitaire malveillant puisse récupérer ces informations depuis la mémoire de l'ordinateur.
Le problème au sein de la machine n'est pas inhérent à MEGA, il concerne toute application de bureau. Si l'hôte est compromis, alors le chiffrement l'est également.
La question du sandboxing est plutôt au sein du navigateur, car même si le standard Javascript est conçu pour être isolé du système, tout dépend de l'implémentation qui en est faite au sein du navigateur.
Sur Firefox par exemple, il a été possible d'aller chercher des données en mémoire au travers de la mémoire partagée X11, il n'est donc pas à exclure que cela puisse se produire avec MEGA.
Au fur et à mesure du temps, la sécurité des navigateurs semble s'améliorer. Cependant, il devient également de plus en plus compliqué de s'assurer de la fiabilité du code en raison de sa quantité.
Afin de réduire ce risque, il convient de ne pas se connecter à MEGA sur une machine infectée ainsi que de ne pas ouvrir d'autres onglets en même temps que MEGA est ouvert et de fermer l'onglet MEGA avant d'utiliser un autre onglet.
Le navigateur détruisant les valeurs en mémoire à la fermeture de l'onglet.

3.5. Certificat compromis

Dans le cas purement théorique où une agence gouvernementale pratiquerait la surveillance de masse, il serait envisageable que le fournisseur de certificat SSL lui délivre un duplicata valide du certificat, cette agence n'aurait donc plus qu'à effectuer du DNS spoofing.

Une telle coopération permettrait logiquement à une agence gouvernementale de faire une attaque man in the middle en se faisant passer pour le client auprès de MEGA et en se faisant passer pour le site pour le client.
Depuis cet instant, il serait possible pour cette agence gouvernementale d'injecter du code Javascript malicieux lui permettant de récupérer les identifiants de l'utilisateur.

3.6. Extension

Dans le but de rendre les transferts de fichier plus rapides, MEGA propose une extension disponible pour les navigateurs internet principaux.
Le 4 septembre 2018, un attaquant a réussi à publier une mise à jour de l'extension sur le Chrome Web Store en tant que MEGA.
Les données des utilisateurs de l'extension Google Chrome se sont vus compromis lors de cet incident.
Le but de cette attaque ne visait même pas MEGA directement, les cibles étaient les identifiants de connexion pour les sites MyEtherWallet, MyMonero, Amazon, GitHub, Google et Microsoft.
L'attaque aurait également pu récupérer les identifiants des utilisateurs MEGA, et donc par conséquent leur clef de chiffrement, donc rendre leur fichiers vulnérables.
Cependant, cet incident met en avant les dangers de passer par un canal de distribution tiers pour des extensions navigateur.

3.7. Robustesse vis-à-vis des autorités

Certains éléments peuvent également poser question, notamment la taille du fichier initial qui sera maintenue à quelques bits près (bits de bourrage/padding).
Prenons le scénario suivant :
Eve est ici un agent du gouvernement ayant un mandat pour enquêter sur les activités de Alice.
Alice stocke le fichier nommé monFichierIllicite.zip dans son espace cloud MEGA.
Bob possède une copie identique du fichier et décide de le mettre en ligne sur son compte à son tour.
Les deux fichiers tels qu'ils sont stockés chez MEGA n'ont aucune similitude, à l'exception d'un détail : les deux fichiers font exactement la même taille, au bit près.
Eve est au courant de la mise en ligne du fichier et exige sous couvert du mandat à MEGA de fournir la liste des utilisateurs ayant mis en ligne ce fichier.
MEGA ne peut bien évidement pas donner cet accès, les fichiers ne contenant pas la même donnée et ayant été chiffrés côté client. MEGA ne peut donc pas fournir cette liste au travers d'une comparaison bit à bit.
Cependant, MEGA connait la taille du fichier mis en ligne par Alice, MEGA peut donc effectuer une recherche par taille de fichiers et cela fera transparaitre le fait que Bob possède également une copie du fichier.
Bien sûr, cette méthode pour s'assurer de la correspondance des deux fichiers n'est pas fiable. Cependant, sur des fichiers de grande taille, il est peu probable d'avoir deux fichiers de même taille au bit près.
Pour éviter ce scénario, il faudrait altérer le fichier mis en ligne. Si Alice et Bob ne mettent pas un fichier de même taille en ligne, il sera plus difficile de rechercher une correspondance entre les deux fichiers.
La méthode la plus simple est d'effectuer une modification au fichier, ajouter du bourrage manuellement sur le fichier. On peut également le compresser ou encore le scinder en plusieurs fichiers.

4. Conclusion

Le service d'hébergement de fichiers MEGA est une alternative solide aux concurrents tels que Dropbox, Microsoft Onedrive ou encore Google Drive.

Le but de la plateforme étant de démocratiser le chiffrement et d'en faire un standard pour l'hébergement de fichiers.
Ce but est atteint, cela transparait dans la simplicité d'utilisation de la plateforme et par conséquent sur le nombre d'utilisateurs, qui s'élève à plus de 230 millions à Q3 2021.
Pour mettre cette information en perspective, le concurrent Dropbox, lui, représente 700 millions d'utilisateurs.
Il n'est pas pertinent de comparer ce chiffre à ceux de Google Drive ou Microsoft OneDrive compte tenu du fait que ces derniers inscrivent automatiquement les utilisateurs de leurs autres produits sur leur service d'hébergement de fichiers.

La sécurité du service est assurée, la qualité du chiffrement est suffisante pour la majorité des utilisations, la faille principale demeure l'utilisateur et sa machine.
La qualité du chiffrement est essentiellement dépendante de l'utilisateur et de son choix de mot de passe.
MEGA ne met pas plus à risque les données d'un utilisateur que les autres plateformes quand ce dernier utilise une machine infectée. Une fois de plus, c'est à l'utilisateur de faire ses choix en matière de sécurité et de les assumer.

Avant d'effectuer les recherches et d'écrire ce rapport, j'avais de gros doutes quant à la sécurité du service. Aujourd'hui, j'hésite à migrer une partie de mes données vers MEGA.
Le fait que le chiffrement se fasse en local et que le code source soit ouvert me rassure, le chiffrement AES étant le chiffrement utilisé par la NSA pour chiffrer les documents secret défense.

5. Bibliographie

https://mega.io/help/client/webclient/general#qu-est-ce-que-mega-veut-dire-57676924886688a70c8b45b5

https://mega.io/privacy

https://mega.nz/help/client/webclient/security-and-privacy#how-does-the-encryption-work-57672896886688a70c8b45ad

https://en.wikipedia.org/wiki/Megaupload

https://mega.io/Mega_Transparency_Report_September_2021.pdf

https://dropbox.gcs-web.com/static-files/1e3e63e5-56e9-4865-8d49-359f23e30189

https://pkware.cachefly.net/webdocs/pkware_pdfs/us_pdfs/white_papers/WP_Entropy-Problem.pdf

https://www.pcmag.com/news/meganz-chrome-extension-hacked-to-steal-logins

https://madaidans-insecurities.github.io/firefox-chromium.html

https://arstechnica.com/information-technology/2013/01/megabad-a-quick-look-at-the-state-of-megas-encryption/

https://github.com/asmcrypto/asmcrypto.js/

https://github.com/meganz/

https://en.wikipedia.org/wiki/DigiNotar

https://en.wikipedia.org/wiki/HMAC

https://mega.nz/SecurityWhitepaper.pdf

https://en.wikipedia.org/wiki/Mega_(service)

https://mega.io/security/bug-bounty

https://businessdesk.co.nz/article/technology/mega-the-encryption-company-that-wants-to-be-everything