Groupe de travail Réseau
Request for Comments : 4978
RFC mise à jour : 3501
Catégorie : sur la voie de la normalisation

A. Gulbrandsen, Oryx Mail Systems GmbH
août 2007

Traduction Claude Brière de L'Isle

Extension IMAP COMPRESS

Résumé
L'extension COMPRESS permet à une connexion IMAP d'être compressée de façon effective et efficace.

Table des Matières

1. Introduction et vue d'ensemble

Un serveur qui prend en charge l'extension COMPRESS l'indique avec un ou plusieurs noms de capacités consistant en "COMPRESS=" suivi par un nom d'algorithme de compression pris en charge comme décrit dans le présent document.

Le but de COMPRESS est de réduire l'utilisation de la bande passante de IMAP.

Comparé à la compression PPP [RFC1962] et à la compression fondée sur le modem (voir [MNP] et [V42BIS]) COMPRESS offre une bien meilleure efficacité de compression. COMPRESS peut être utilisé avec la sécurité de la couche transport (TLS, *Transport Security Layer*) [RFC4346], avec le chiffrement d'authentification simple et couche de sécurité (SASL, *Simple Authentication and Security layer*) avec des réseaux privés virtuels (VPN, *Virtual Private Network*) etc. Comparé à la compression TLS [RFC3749], COMPRESS présente les avantages (inconvénients) suivants :

- COMPRESS peut être mis en œuvre facilement par les serveurs et clients IMAP.

- IMAP COMPRESS bénéficie d'une connaissance intime de l'automate à états du protocole IMAP, ce qui permet une optimisation dynamique et agressive des paramètres de l'algorithme de compression sous-jacent.

- Quand la couche TLS met en œuvre la compression, tout protocole qui utilise cette couche peut bénéficier en toute transparence de cette compression (par exemple, SMTP et IMAP). COMPRESS est spécifique de IMAP.

Afin d'augmenter l'interfonctionnement, il est souhaitable d'avoir aussi peu d'algorithmes de compression différents que possible, c'est pourquoi le présent document n'en spécifie qu'un. L'algorithme DEFLATE (défini dans la [RFC1951]) est un standard, largement disponible et très efficace, et il est donc le seul algorithme défini dans ce document.

Afin d'augmenter l'interopérabilité, les serveurs IMAP qui annoncent cette extension DEVRAIENT aussi annoncer le mécanisme de compression TLS DEFLATE comme défini dans la [RFC3749]. Les clients IMAP PEUVENT utiliser la compression COMPRESS ou TLS, cependant, si le client et le serveur prennent en charge les deux, il est RECOMMANDÉ que le client choisisse la compression TLS.

L'extension ajoute une nouvelle commande (COMPRESS) et pas de nouvelle réponse.

2. Conventions utilisées dans le document

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

La syntaxe formelle est définie par la [RFC4234] telle que modifiée par la [RFC3501].

Dans les exemples, "C:" et "S:" indiquent les lignes envoyées respectivement par le client et le serveur. "[...]" note une élision.

3. Commande COMPRESS

Arguments : Nom du mécanisme de compression : "DEFLATE".

Réponse : aucune

Résultat :
OK : le serveur va compresser ses réponses et s'attend à ce que le client compresse ses commandes.
NO : la compression est déjà active via une autre couche.
BAD : commande inconnue, argument invalide ou inconnu, ou COMPRESS déjà actif.

La commande COMPRESS donne pour instruction au serveur d'utiliser le mécanisme de compression désigné ("DEFLATE" est le seul défini) pour toutes les commandes et/ou réponses après COMPRESS.

Le client NE DOIT PAS envoyer d'autre commande avant d'avoir vu le résultat de COMPRESS. Si la réponse était OK, le client DOIT compresser en commençant par la première commande après COMPRESS. Si la réponse du serveur était BAD ou NO, le client NE DOIT PAS activer la compression.

Si le serveur répond NO parce que il sait que le même mécanisme est déjà actif (par exemple, parce que TLS a négocié le même mécanisme) il DOIT envoyer COMPRESSIONACTIVE comme code de texte de réponse (voir au paragraphe 7.1 de la [RFC3501]) et le texte de réponse DEVRAIT dire quelle couche compresse.

Si le serveur produit une réponse OK, le serveur DOIT compresser en commençant immédiatement après le CRLF qui termine la réponse OK étiquetée. (Les réponses produites par le serveur avant la réponse OK vont bien sûr rester non compressées.) Si le serveur produit une réponse BAD ou NO, il NE DOIT PAS activer la compression.

Pour DEFLATE (comme pour de nombreux autres mécanismes de compression) le compresseur peut faire un compormis entre vitesse et qualité. À la décompression, il n'y a pas matière à compromis. Par conséquent, le client et le serveur sont tous deux libres de prendre le meilleur taux raisonnable de compression pour les données qu'ils envoient.

Quand COMPRESS est combiné avec les couches de sécurité TLS (voir la [RFC4346]) ou SASL (voir la [RFC4422]) l'ordre d'envoi des trois extensions DOIT être d'abord COMPRESS, puis SASL, et enfin TLS. C'est-à-dire qu'avant que les données soient transmises, elles sont d'abord compressées. Ensuite, si une couche de sécurité SASL a été négociée, les données compressées sont alors signées et/ou chiffrées en conséquence. Enfin, si une couche de sécurité TLS a été

négociée, les données provenant de l'étape précédente sont signées et/ou chiffrées en conséquence. À la réception des données, l'ordre de traitement DOIT être inversé. Cela assure qu'avant l'envoi, les données sont compressées avant d'être chiffrées, indépendamment de l'ordre dans lequel le client produit les COMPRESS, AUTHENTICATE, STARTTLS.

L'exemple suivant illustre comment les commandes et réponses sont compressées durant une simple séquence d'enregistrement :

S: * OK [CAPABILITY IMAP4REV1 STARTTLS COMPRESS=DEFLATE]
C: a starttls
S: a OK TLS active

À partir de ce point, tout est chiffré.

C: b login arnt tnra
S: b OK Logged in as arnt
C: c compress deflate
S: c OK DEFLATE active

À partir de ce point, tout est compressé avant d'être chiffré.

L'exemple suivant montre comment un serveur peut refuser de compresser deux fois :

S: * OK [CAPABILITY IMAP4REV1 STARTTLS COMPRESS=DEFLATE]
[...]
C: c compress deflate
S: c NO [COMPRESSIONACTIVE] DEFLATE active via TLS

4. Efficacité de compression

Cette section est pour information, elle n'est pas normative.

IMAP pose des problème inhabituels à une couche de compression.

L'amont est très simple. La plupart des clients IMAP envoient encore et encore toujours les mêmes quelques commandes, de sorte que tout algorithme de compression qui peut exploiter la répétition fonctionne efficacement. La commande APPEND est une exception ; les clients qui envoient de nombreuses commandes APPEND peuvent vouloir entourer de grands littéraux avec des couleurs de la même façon que ce qui est recommandé pour les serveurs un peu plus loin dans cette section.

L'aval a la propriété inhabituelle que plusieurs sortes de données sont envoyées, plongeant dans la confusion tous les algorithmes de compression fondés sur des dictionnaires.

Un type est celui des réponses IMAP. Elles sont très compressibles ; zlib utilisant son moindre réglage d'intensité de CPU compresse les réponses normales à entre 25 et 40 % de leur taille d'origine.

Un autre type est celui des en-têtes de messagerie électronique. Ils sont également compressibles, et bénéficient de l'utilisation du même dictionnaire que les réponses IMAP.

Un troisième type est le texte de corps de messagerie électronique. Le texte est généralement très court et comporte beaucoup d'ASCII, de sorte que le même dictionnaire de compression va faire là aussi du bon travail. Quand plusieurs messages dans la même trame sont lus au même moment, des lignes entre guillements etc., peuvent souvent être compressés presque à zéro.

Finalement, les pièces jointes (corps de messagerie électronique non textuels) sont transmises, soit en forme binaire, soit codées en base-64.

Quand les pièces jointes sont restituées en forme binaire, DEFLATE peut être capable de les compresser, mais le format de la pièce jointe n'est généralement pas de type IMAP, de sorte que le dictionnaire construit lors de la compression de IMAP ne va pas aider. Le compresseur doit adapter son dictionnaire IMAP au format de la pièce jointe, et vice-versa. Certains

formats de fichier ne sont pas compressibles du tout en utilisant deflate, par exemple, les fichiers .gz, .zip, et .jpg.

Quand des pièces jointes sont restituées en forme base-64, les mêmes problèmes s'appliquent, mais le codage base-64 ajoute un autre problème. Les algorithmes de compression sur 8 bits comme deflate fonctionnent bien sur des formats de fichier de 8 bits, cependant le base-64 retourne un fichier en quelque chose qui ressemble à des octets de 6 bits, cachant la plus grande partie du format de fichier de 8 bits au compresseur.

Quand on utilise la bibliothèque zlib (voir la [RFC1951]) les fonctions deflateInit2(), deflate(), inflateInit2(), et inflate() suffisent à mettre en œuvre cette extension. La valeur windowBits doit être dans la gamme de -8 à -15, ou alors deflateInit2() utilise le mauvais format. deflateParams() peut être utilisé pour améliorer le taux de compression et l'utilisation des ressources. L'argument Z_FULL_FLUSH sur deflate() peut être utilisé pour purger le dictionnaire (l'homologue receveur n'a pas besoin de faire quelque chose).

Un client peut améliorer la compression en aval en mettant en œuvre BINARY (défini dans la [RFC3516]) et en utilisant FETCH BINARY à la place de FETCH BODY. À la connaissance de l'auteur, l'amélioration va de 5 % à 40 % selon la pièce jointe à télécharger.

Un serveur peut améliorer la compression en aval si il donne au compresseur l'indication que le type de données va fortement changer, par exemple, en envoyant un Z_FULL_FLUSH au début et à la fin de grands littéraux non textuels (avant et après '*CHAR8' dans la définition de "literal" dans la RFC 3501, page 86). Il vaut mieux laisser les petits littéraux comme ils sont. Une limite possible est 5 k.

Un serveur peut améliorer l'efficacité de la CPU chez le serveur et le client si il ajuste le niveau de compression (par exemple, en utilisant la fonction deflateParams() dans zlib) à ce moment, pour éviter d'essayer de compresser des pièces jointes incompressibles. Une stratégie très simple est de changer le niveau à 0 au début d'un littéral pourvu que les deux premiers octets soient 0x1F 0x8B (comme dans des fichiers deflate-compressed) ou 0xFF 0xD8 (JPEG), et de le garder à 1-5 le reste du temps. Des stratégies plus complexes sont possibles.

5. Syntaxe formelle

La spécification de syntaxe qui suit utilise la notation de forme Backus-Naur augmenté (ABNF) comme spécifiée dans la [RFC4234]. Cette syntaxe s'ajoute à la grammaire spécifiée dans la [RFC3501]. La [RFC4234] définit SP et la [RFC3501] définit command-auth, capability, et resp-text-code.

Sauf mention contraire, tous les caractères alphabétiques sont insensibles à la casse. L'utilisation de caractères majuscules ou minuscules pour définir des chaînes de jetons est seulement pour facilité la lisibilité. Les mises en œuvre DOIVENT accepter ces chaînes de façon insensible à la casse.

```
command-auth =/ compress

compress = "COMPRESS" SP algorithm

capability =/ "COMPRESS=" algorithm          ;; plusieurs capacités COMPRESS sont permises.

algorithm = "DEFLATE"

resp-text-code =/ "COMPRESSIONACTIVE"
```

Noter qu'à cause de la syntaxe des noms de capacités, les futurs noms d'algorithmes doivent être des atomes.

6. Considérations sur la sécurité

Comme pour la compression TLS [RFC3749].

7. Considérations relatives à l'IANA

L'IANA a ajouté COMPRESS=DEFLATE à la liste des capcités IMAP.

8. Remerciements

Eric Burger, Dave Cridland, Tony Finch, Ned Freed, Philip Guenther, Randall Gellens, Tony Hansen, Cullen Jennings, Stephane Maes, Alexey Melnikov, Lyndon Nerenberg,et Zoltan Ordogh ont tous aidé au présent document.

L'auteur tient aussi à remercier les diverses personnes qui dans les réunions ont apporté une aide réelle, mais ne figurent pas dans la boîte aux lettres de l'auteur.

9. Références

9.1 Références normatives

[RFC1951] P. Deutsch, "Spécification du format DEFLATE de données compressées, version 1.3", mai 1996.

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC3501] M. Crispin, "Protocole d'accès au message Internet - version 4rev1", mars 2003. *(P.S. ; MàJ par* RFC4466, 4469, 4551, 5032, 5182, 7817, 8314, 8437, 8474 ; *remplacée par la* RFC9051

[RFC4234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", octobre 2005. *(Remplace* RFC2234, *remplacée par* RFC5234)

9.2 Références pour information

[MNP] Gilbert Held, "The Complete Modem Reference", Second Edition, Wiley Professional Computing, ISBN 0-471-00852-4, mai 1994.

[RFC1962] D. Rand, "Protocole de contrôle de compression en PPP (CCP)", juin 1996.

[RFC3516] L. Nerenberg, "Extension Contenu binaire à IMAP4", avril 2003. (*MàJ par* RFC4466) *(P.S.)*

[RFC3749] S. Hollenbeck, "Méthodes de compression du protocole de sécurité de la couche transport", mai 2004. *(P.S.)*

[RFC4346] T. Dierks et E. Rescorla, "Protocole de sécurité de la couche Transport (TLS) version 1.1", avril 2006. *(Remplace* RFC2246 *; Remplacée par* RFC5246 *; MàJ par* RFC4366, 4680, 4681, 5746, 6176, 7465, 7507, 7919)

[RFC4422] A. Melnikov et K. Zeilenga, éd, "Authentification simple et couche de sécurité (SASL)", juin 2006. *(P.S.)*

[V42BIS] Recommandation UIT-T V.42bis, "Procédures de compression de données pour équipements de terminaison de circuits de données (DCE) utilisant des procédures de correction d'erreur", http://www.itu.int/rec/T-REC-V.42bis, janvier 1990.

Adresse de l'auteur

Arnt Gulbrandsen
Oryx Mail Systems GmbH
Schweppermannstr. 8
D-81671 Muenchen

Germany

Fax : +49 89 4502 9758
mél : arnt@oryx.com

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.