Groupe de travail Réseau
Request for Comments : 4217
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

P. Ford-Hutchinson
IBM UK Ltd

octobre 2005

Sécurisation de FTP avec TLS

Résumé
Le présent document décrit un mécanisme qui peut être utilisé par les clients et serveurs FTP pour mettre en œuvre la sécurité et l'authentification en utilisant le protocole TLS défini par la RFC 2246, "Protocole TLS version 1.0.", et les extensions au protocole FTP définies par la RFC 2228, "Extensions à la sécurité de FTP". Il décrit le sous ensemble des extensions qui est exigé et les paramètres à utiliser, discute certaines des questions de politique que clients et serveurs vont devoir résoudre, considère certaines des implications de ces politiques, et discute des comportements attendus de la part des mises en œuvre pour permettre l'interfonctionnement. Le présent document est destiné à fournir à TLS la prise en charge de FTP de façon similaire à celle fournie pour SMTP dans la RFC 2487, "Extension de service SMTP pour un SMTP sûr sur la sécurité de la couche Transport", et pour HTTP dans la RFC 2817, "Mise à niveau de TLS au sein de HTTP/1.1.".

La présente spécification est en accord avec la RFC 959, "Protocole de transfert de fichier". Elle s'appuie sur la RFC 2246, "Protocole TLS version 1.0.", et la RFC 2228, "Extensions de sécurité à FTP".

Table des matières

1. Introduction

Le présent document décrit comment trois autre documents devraient être combinés pour fournir un protocole de transfert de fichiers utile, interopérable, et sûr. Ces documents sont :
la [RFC0959] : description du protocole de transfert de fichiers sur l'Internet,
la [RFC2246] : description du protocole de sécurité de la couche Transport (développé à partir de la version 3,0 du protocole de couche de prises sûres de Netscape (SSL, *Secure Sockets Layer*),
la [RFC2228] : extensions au protocole FTP pour permettre la négociation de mécanismes de sécurité assurant l'authentification, la confidentialité, et l'intégrité du message.

Le présent document est destiné à fournir la prise en charge de TLS pour FTP de la même façon que celle fournie pour SMTP dans la [RFC3207] et HTTP dans la [RFC2817].

Les extensions de sécurité à FTP dans la [RFC2228] offrent un ensemble complet des commandes et de réponses qui peuvent être utilisées pour ajouter l'authentification, l'intégrité, et la confidentialité au protocole FTP. Le protocole TLS est un mécanisme populaire (dû à son adoption générale dans l'environnement HTTP) pour sécuriser généralement une connexion de prise.

Bien que TLS ne soit pas le seul mécanisme pour sécuriser un transfert de fichier, il offre certains des attributs positifs suivants :
- Niveaux de sécurité flexibles : TLS peut prendre en charge la confidentialité, l'intégrité, l'authentification, ou leur combinaison. Durant une session, cela permet aux clients et serveurs de décider de façon dynamique du niveau de sécurité requis pour un transfert de données particulier.
- Capacité à fournir une forte authentification du serveur FTP.
- Il est possible d'utiliser les identités TLS pour authentifier les utilisateurs et hôtes clients.
- Gestion de clé publique formalisée : par l'utilisation de mécanismes d'identification de client bien établis (pris en charge par TLS) durant la phase d'authentification, la gestion de certificat peut être construite dans une fonction centrale. Bien que ce ne soit pas désirable pour toutes les utilisations de transfert de fichier sécurisé, elle offre des avantages dans certains environnements structurés.
- Coexistence et interfonctionnement avec les mécanismes d'authentification qui sont déjà en place pour le protocole HTTPS. Cela permet aux navigateurs de la Toile d'incorporer le transfert de fichier sûr en utilisant la même infrastructure qu'établie pour permettre la navigation sûre sur la Toile.

Le protocole TLS est un développement du protocole SSL de la Netscape Communication Corporation et le présent document peut être utilisé pour permettre au protocole FTP d'être utilisé avec SSL ou TLS. Le protocole utilisé réel sera décidé par la négociation de la session protégée par la couche TLS/SSL. Le présent document se référera seulement au protocole TLS ; cependant, il est entendu que le client et le serveur PEUVENT en fait utiliser SSL si ils sont configurés à le faire.

Ces trois protocoles peuvent être combinés de nombreuses façons différentes. Le présent document choisit une méthode par laquelle FTP peut opérer en toute sécurité, tout en fournissant à la fois flexibilité et interopérabilité. Cela nécessite une brève description du mécanisme de négociation réel, une description détaillée des politiques et pratiques requises, et une

discussion des comportements attendus des clients et serveurs pour permettre à l'une et l'autre partie d'imposer ses exigences de sécurité dans la session FTP.

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].

2. Public visé

Le présent document s'adresse aux développeurs qui souhaitent mettre en œuvre TLS comme mécanisme de sécurité pour des clients et/ou serveurs FTP sûrs.

Les administrateurs et architectes de systèmes devraient être pleinement conscients des implications de sécurité discutées dans la [RFC2228], qui doivent être considérées lors du choix d'une mise en œuvre de ce protocole et de sa configuration pour fournir la sécurité requise.

3. Généralités

Une description complète des améliorations au protocole de sécurité de FTP est contenue dans la [RFC2228]. Ce document décrit comment les commandes AUTH, PROT, PBSZ, et CCC, qui y sont définies, devraient être mises en œuvre avec le protocole TLS.

En résumé, une session FTP est établie sur l'accès de contrôle normal. Un client demande TLS avec la commande AUTH et décide ensuite si il souhaite sécuriser les connexions de données par l'utilisation des commandes PBSZ et PROT. Si un client souhaite que la connexion de contrôle revienne au texte source (par exemple, une fois que la phase d'authentification est achevée) la commande CCC peut alors être utilisée.

La mise en œuvre de cette extension de protocole n’assure pas que chaque session et transfert de données est sécurisé ; elle donne simplement les outils qui permettent à un client et/ou serveur de négocier un niveau acceptable ou exigé de sécurité pour la session ou transfert de données en cause. Cependant, il est possible qu'une mise en œuvre de serveur qui est capable de refuser fonctionne de façon non sûre.

4. Négociation de session sur l'accès de commande

Le serveur écoute sur l'accès de contrôle FTP normal {FTP-PORT} et l'initiation de session n'est pas sécurisée du tout. Une fois que le client souhaite sécuriser la session, la commande AUTH est envoyée et le serveur PEUT alors permettre que la négociation de TLS ait lieu.

4.1 Le client veut une session sécurisée

Si un client souhaite tenter de sécuriser une session, il DEVRAIT alors, conformément à la [RFC2228], envoyer la commande AUTH avec le paramètre de demande de TLS {TLS-PARM} ('TLS').

Le client doit alors se comporter en accord avec sa politique selon la réponse reçue du serveur et aussi le résultat de la négociation TLS. Un client qui reçoit un rejet de AUTH PEUT choisir de continuer avec la session non protégée si il le désire.

4.2 Le serveur veut une session sécurisée

Le protocole FTP ne permet pas à un serveur d'imposer directement un comportement au client ; cependant, le même effet peut être obtenu en refusant d'accepter certaines commandes FTP jusqu'à ce que la session soit sécurisée à un niveau acceptable pour le serveur.

Dans l'un et l'autre cas, "234" est la réponse du serveur à une commande "AUTH TLS" qu'il va honorer.

La réponse "334", comme définie dans la [RFC2228], implique qu'un échange ADAT suive. Le présent document n'utilise pas la commande ADAT et donc la réponse "334" est incorrecte.

Le protocole FTP insiste pour qu'une commande USER soit utilisée pour identifier l'entité qui tente d'utiliser le serveur ftp. Bien que la négociation TLS puisse fournir des informations d'authentification, la commande USER DOIT quand même être produite par le client. Cependant, ce sera une question de mise en œuvre de serveur de décider quels accréditifs accepter et quelles vérifications de cohérence appliquer entre le certificat de client utilisé et le paramètre sur la commande USER.

La [RFC2228] déclare que l'utilisateur doit se réautoriser (c'est-à-dire, produire à nouveau une ou toutes commande USER, PASS, et ACCT) suite à la commande AUTH. De plus, le présent document spécifie que tous les autres paramètres de transfert (autres que le paramètre AUTH) doivent être réinitialisés, presque comme si une commande REIN était produite.

Réinitialisation des paramètres de transfert après la commande AUTH, incluant (mais sans s'y limiter) : identité d'utilisateur, accès de données par défaut, TYPE, STRU, MODE, et répertoire de travail courant.

5. Suppression de l'accès de contrôle

Il y a des circonstances dans lesquelles il peut être souhaitable de ne protéger la connexion de contrôle que durant une partie de la session et ensuite de revenir à une connexion en clair. C'est souvent dû aux limitations d'appareils frontières comme les NAT et pare-feu, qui s'attendent à être capables d'examiner le contenu de la connexion de contrôle afin de modifier leur comportement.

Normalement, les commandes AUTH, USER, PASS, PBSZ, et PROT vont être protégées au sein du protocole TLS et ensuite, la commande CCC va être produite pour retourner à un état de prise de texte en clair. Ceci pose d'importants problèmes de sécurité (qui sont discutés dans la section Considérations sur la sécurité) mais le présent document décrit comment la commande devrait être utilisée, si le client et le serveur souhaitent toujours l'utiliser après avoir considéré le produit.

Quand un serveur reçoit la commande CCC, il devrait se comporter comme suit :
Si le serveur n'accepte pas les commandes CCC (ou ne les comprend pas) une réponse 500 devrait être envoyée.
Autrement, si la connexion de contrôle n'est pas protégée par TLS, une réponse 533 devrait alors être envoyée.
Autrement, si le serveur ne souhaite pas permettre que la connexion de contrôle soit supprimée pour l'instant, une réponse 534 devrait alors être envoyée.
Autrement, le serveur accepte la commande CCC et devrait faire ce qui suit :

- envoyer une réponse 200 ;
- fermer la session TLS sur la prise et la laisser ouverte ;
- continuer la connexion de contrôle en clair, en s'attendant à ce que la prochaine commande du client soit en clair ;
- ne plus accepter d'autre commande PBSZ ou PROT. Tous les transferts de données suivants doivent être protégés avec les réglages PROT courants.

6. Réponse à la commande FEAT

La commande FEAT (introduite dans la [RFC2389]) permet aux serveurs qui ont des caractéristiques supplémentaires de les annoncer à un client en répondant à la commande FEAT. Si un serveur prend en charge la commande FEAT, il DOIT alors annoncer les commandes AUTH, PBSZ, et PROT prises en charge dans la réponse, comme décrit au paragraphe 3.2 de la [RFC2389]. De plus, la commande AUTH devrait avoir une réponse qui identifie "TLS" comme un des paramètres possibles de AUTH. Il n'est pas nécessaire d'identifier le synonyme "TLS-C" séparément.

Exemple de réponse (dans le même style que la [RFC2389])

C> FEAT
S> 211 Extensions prises en charge
S> AUTH TLS
S> PBSZ
S> PROT
S> 211 FIN

7. Comportement de la connexion de données

Dans le modèle FTP, la connexion de données peut être utilisée d'une des trois façons suivantes. (Note : ces descriptions ne sont pas nécessairement placées dans l'exact ordre chronologique, mais elles décrivent bien les étapes requises. Voir plus loin les diagrammes qui les précisent.)

i) Échange classique de données entre client/serveur FTP
   - Le client obtient un accès, envoie le numéro d'accès au serveur, le serveur connecte le client. Le client produit une demande d'envoi ou de réception au serveur sur la connexion de contrôle et le transfert de données commence sur la connexion de données.

ii) Échange de données client/serveur par un pare-feu amical (comme discuté dans la [RFC1579]) en utilisant la commande PASV pour inverser la direction de la connexion de données.
   - Le client demande que le serveur ouvre un accès, le serveur obtient un accès et retourne l'adresse et le numéro d'accès au client, le client se connecte au serveur sur cet accès. Le client produit une commande d'envoi ou de réception sur la connexion de contrôle ; le transfert de données commence sur la connexion de données.

iii) Échange de données client/serveur initié par le client (mandataire ou connexions PASV).
   - Le client demande que le serveur A ouvre un accès ; le serveur A obtient un accès et le retourne au client ; le client envoie ce numéro d'accès au serveur B. Le serveur B se connecte au serveur A. Le client envoie une demande d'envoi ou de réception au serveur A et la fait suivre au serveur B et le transfert de données commence. Dans ce modèle, le serveur A est le mandataire ou hôte PASV et est un client pour la connexion de données au serveur B.

Pour i) et ii), le client FTP DOIT être le client TLS et le serveur FTP DOIT être le serveur TLS.

C'est-à-dire que le côté qui initie la connexion avec un appel connect() ou le côté qui réagit à la connexion via l'appel accept() est sans importance ; le client FTP, comme défini dans la [RFC0959], est toujours le client TLS, comme défini dans la [RFC2246].

Dans le scénario iii), il y a un problème en ce que ni le serveur A ni le serveur B n'est le client TLS, étant donné qu'un serveur FTP doit agir comme un serveur TLS pour le FTP amical aux pare-feu [RFC1579]. Donc, ceci est explicitement exclu dans le document des extensions de sécurité [RFC 2228] et dans le présent document.

8. Mécanismes de la commande AUTH

La commande AUTH prend un seul paramètre pour définir le mécanisme de sécurité à négocier. Comme les protocoles SSL/TLS auto négocient leurs niveaux, il n'est pas besoin de distinguer entre SSL et TLS à la couche d'application. Le nom du mécanisme pour négocier TLS est la chaîne de caractères identifiée dans {TLS-PARM}. Cela permet au client et au serveur de négocier TLS sur la connexion de contrôle sans altérer la protection du canal de données. Pour protéger aussi le canal de données, la séquence de commande PBSZ suivie par la commande PROT DOIT être utilisée.

Note : l'état de la connexion de données PEUT être modifié par le client qui produit la commande PROT avec le nouveau niveau désiré de protection du canal de données et le serveur qui répond par l'affirmative. Cette négociation de la protection du canal de données peut se produire en tout point de la session (même juste après une commande PORT ou PASV) et aussi souvent que nécessaire. Voir aussi à la Section 16, "Considérations relatives à l'IANA".

9. Sécurité de la connexion de données

Le niveau de sécurité de la connexion de données est déterminé par la commande PROT.
La commande PROT, comme spécifiée dans la [RFC2228], permet la négociation client/serveur du niveau de sécurité de la connexion de données. Une fois qu'une commande PROT a été produite par le client et acceptée par la retour d'une réponse "200" du serveur, la sécurité des connexions de données suivantes DOIT être à ce niveau jusqu'à ce qu'une autre commande PROT soit produite et acceptée ; la session se termine et une commande REIN est produite, ou la sécurité de la session est renégociée (via une commande AUTH).

Négociation de la sécurité de la connexion de données (avec la commande PROT).
Note : conformément à la [RFC2228], il n'y a pas de facilité pour sécuriser la connexion de données avec une connexion de contrôle non sûre. Précisément, la commande PROT DOIT être précédée d'une commande PBSZ, et une commande PBSZ

DOIT être précédée d'un échange réussi de données de sécurité (la négociation TLS dans ce cas). La commande définie dans la [RFC2228] pour négocier la sécurité de la connexion de données est la commande PROT. Comme elle est définie, il y a quatre valeurs que le paramètre de commande PROT peut prendre :
'C' (*Clear)* - ni intégrité ni confidentialité
'S' *(Safe)* - intégrité sans confidentialité
'E' - confidentialité sans intégrité
'P' *(Private)* -intégrité et confidentialité

Comme la négociation TLS englobe (et excède) la distinction Safe /Confidential / Private, seules les valeurs Private (utilisation de TLS) et Clear (ne pas utiliser TLS) sont utilisées.

Pour TLS, la connexion de données peut avoir un des deux niveaux de sécurité :
1) 'C' (demandé par 'PROT C')
2) 'P' (demandé par 'PROT P')

Avec le niveau de protection 'C', la connexion de données est faite sans TLS. Donc, la connexion n'est pas authentifiée et n'a pas de protection de la confidentialité ou de l'intégrité. Cela peut être le comportement désiré pour les serveurs qui envoient des listes de fichiers, des données préalablement chiffrées, ou des données non sensibles (par exemple, pour les serveurs de FTP anonyme).

Si le niveau de sécurité de la connexion de données est 'P', une négociation TLS doit alors avoir lieu sur la connexion de données pour la satisfaction du client et du serveur avant qu'aucune donnée puisse être transmise sur la connexion. Les couches TLS du client et du serveur seront chargées de négocier les suites de chiffrement TLS exactes qui seront utilisées (et donc l'éventuelle sécurité de la connexion).

De plus, la commande Taille de mémoire tampon de protection (PBSZ, *protection buffer size*) comme précisée dans la [RFC2228], est obligatoire avant toute commande PROT. Le présent document définit aussi un mécanisme d'encapsulation de canal de données pour les mémoires tampon de données protégées. Pour FTP-TLS, qui apparaît à l'application FTP comme un mécanisme de protection de flux, ceci n'est pas exigé. Donc, la commande PBSZ DOIT quand même être produite, mais doit avoir un paramètre de "0" pour indiquer qu'aucune mise en mémoire tampon n'a lieu et que la connexion de données ne devrait pas être encapsulée.

Noter que PBSZ 0 n'est pas dans la grammaire du paragraphe 8.1 de la [RFC2228], où il est dit :

PBSZ <sp> <entier décimal> <CRLF> <entier décimal> ::= tout entier décimal de 1 à (2^32)-1

Cependant, on notera que l'utilisation d'une valeur de "0" pour signifier un protocole de flux est une utilisation raisonnable de "0" pour ce paramètre et n'est pas ambiguë.

Sécurité initiale de la connexion de données
L'état initial de la connexion de données DOIT être 'C' (c'est le comportement indiqué par la [RFC2228]).

10. Discussion du comportement de négociation

Comme la [RFC2228] permet de négocier, activer et désactiver de façon dynamique les qualités de la sécurité, cela peut faire que les mises en œuvre peuvent sembler assez complexes. Cependant, dans toute instance donnée, le comportement devrait être assez direct. Soit le serveur va appliquer la politique de l'hôte serveur, soit il va fournir les capacités de sécurité demandées par le client. Soit le client se conforme à la politique du serveur, soit il entreprend de fournir les capacités que l'utilisateur désire.

10.1 Connexion de contrôle vue du serveur

Un serveur PEUT avoir une déclaration de politique qui pourrait :
- refuser toute commande avant que TLS soit négocié (cela peut causer des problèmes si une commande SITE ou autre équivalente est requise avant l'établissement) ;
- refuser certaines commandes avant que TLS soit négocié (par exemple, USER, PASS, ou ACCT) ;
- refuser des commandes USER non sûres provenant de certains utilisateurs (par exemple, pas de ftp/anonymous) ;
- refuser toutes commandes USER sûres pour certains utilisateurs (par exemple, ftp/anonymous) ;
- définir le ou les niveaux de TLS à permettre ;
- définir les suites de chiffrement permises (peut-être sur la base de l'hôte/domaine/...) ;

- permettre l'authentification TLS comme substitut de l'authentification locale.
- définir les politiques de connexion de données (voir au paragraphe suivant).

Il est possible que la négociation TLS ne soit pas complètement satisfaisante pour le serveur, et dans ce cas, il peut être dans un des états suivants :

La négociation TLS a complètement échoué :
Dans ce cas, la connexion de contrôle devrait être encore dans un mode non protégé et le serveur DEVRAIT produire une réponse non protégée "421" pour terminer la session.

La négociation TLS s'est achevée avec succès, mais le serveur décide que les paramètres de session ne sont pas acceptables (par exemple, il n'est pas permis au serveur d'utiliser le nom distinctif dans le certificat du client). Dans ce cas, la connexion de contrôle devrait être quand même dans un état protégé, de sorte que le serveur PEUT soit continuer de refuser de servir la commande, soit produire une réponse protégée "421" et clore la connexion.

La négociation TLS a échoué durant la prise de contact TLS : dans ce cas, la connexion de contrôle est dans un état inconnu et le serveur DEVRAIT simplement éliminer la connexion de contrôle.

Le code du serveur sera chargé de mettre en œuvre les politiques requises et de s'assurer que le client est empêché de circonvenir la sécurité choisie en refusant de servir les commandes qui sont contraires à sa politique.

10.2 Connexion de données vue du serveur

Le serveur peut prendre une des quatre vues de base de la connexion de données.
1 - Ne pas permettre du tout le chiffrement (dans ce cas, la commande PROT ne devrait permettre aucune valeur autre que 'C' - si il est permis).
2 - Permettre au client de choisir la protection ou non.
3 - Insister sur la protection des données (dans ce cas, la commande PROT doit être produite avant la première tentative de transfert de données).
4 - Décide un des trois ci-dessus pour chaque connexion de données.

Le serveur DEVRAIT seulement vérifier l'état du niveau de protection des données (pour les options 3 et 4 ci-dessus) sur la commande réelle qui va initier le transfert de données (et non sur le PORT ou PASV). Les commandes suivantes, définies dans la [RFC0959], causent l'ouverture des connexions de données et donc peuvent être rejetées avant tout message 1xx à cause d'un réglage incorrect de PROT.

STOR
RETR
NLST
LIST
STOU
APPE

La réponse pour indiquer que le réglage de PROT est incorrect est "521, la connexion de données ne peut pas être ouverte avec ce réglage de PROT".

Si le niveau de protection indique que TLS est exigé, il devrait alors être négocié une fois la connexion de donnée établie. Donc, la réponse "150" déclare seulement que la commande peut être utilisée avec le niveau PROT actuel. Si le serveur n'aime pas la négociation TLS, il va alors clore immédiatement l'accès de données et faire suivre la commande "150" d'une réponse "522", qui indique que la négociation TLS a échoué ou était inacceptable. (Note : Cela signifie que l'application peut passer une liste standard de suites de chiffrement à la couche TLS pour la négociation, et revoir celle négociée pour son applicabilité dans chaque instance).

La section Considérations sur la sécurité discute des problèmes de vérification croisée de tout certificat utilisé pour authentifier la connexion de données avec celui ou ceux utilisés pour l'authentification de la connexion de contrôle. C'est une étape importante de la sécurité.

Il est raisonnable que le serveur insiste pour que la connexion de données utilise une session TLS en antémémoire. Ce peut être une antémémoire d'une connexion de données antérieure sur une connexion de contrôle supprimée. Si c'est la raison pour refuser de permettre le transfert des données, la réponse "522" devrait alors l'indiquer.

Note : ceci a un impact important sur la conception du client, mais permet aux serveurs de minimiser les cycles utilisés durant la négociation TLS en refusant d'effectuer une pleine négociation avec un client antérieurement authentifié.

On notera que l'authentification TLS du serveur sera l'authentification de l'hôte serveur lui-même et non d'un utilisateur sur l'hôte serveur.

10.3 Connexion de contrôle vue du client

Dans la plupart des cas, il est probable que le client va utiliser TLS parce que le serveur refuserait une interaction non sécurisée. Pour le permettre, les clients DEVRAIENT être assez souples pour gérer la sécurisation d'une session au moment approprié et permettre quand même que les politiques d'utilisateur/serveur dictent exactement quand durant la session est négociée la sécurité.

Dans le cas où c'est le client qui insiste sur la sécurisation de la session, le client va devoir s'assurer que les négociations se sont toutes terminées de façon satisfaisante et devra être capable d'informer l'utilisateur de façon intelligente si le serveur ne prend pas en charge, ou n'est pas prêt à utiliser, les niveaux de sécurité requis.

Les clients DEVRAIENT être codés d'une manière telle qu'elle permette un étagement souple dans le temps des commandes AUTH, PBSZ, et PROT qui soit dicté par le serveur. Il est assez raisonnable qu'un serveur refuse certaines commandes avant que ces commandes soient reçues. De même, il est possible qu'une commande SITE ou autre soit nécessaire à un serveur avant le AUTH. Un client DOIT permettre à un utilisateur d'outrepasser l'ordre temporel de ces commandes pour convenir à un serveur spécifique.

Par exemple, un client NE DEVRAIT PAS insister pour envoyer AUTH comme première commande dans une session, ni insister pour produire directement une paire PBSZ/PROT après le AUTH. Cela peut bien être le comportement par défaut, mais doit pouvoir être outrepassé par un utilisateur.

La négociation TLS peut n'être pas complètement satisfaisante pour le client, auquel cas, il va être dans un des états suivants :

La négociation TLS a complètement échoué : dans ce cas, la connexion de contrôle devrait quand même être dans un mode non protégé et le client devrait produire une commande QUIT non protégée pour terminer la session.

La négociation TLS s'est achevée avec succès, mais le client décide que les paramètres de session ne sont pas acceptables (par exemple, le nom distinctif dans le certificat n'est pas le vrai serveur attendu). Dans ce cas, la connexion de contrôle devrait quand même être active dans un mode protégé, de sorte que le client devrait produire une commande QUIT protégée pour terminer la session.

La négociation TLS a échoué durant la prise de contact TLS : dans ce cas, la connexion de contrôle est dans un état inconnu et le client devrait simplement abandonner la connexion de contrôle.

10.4 Connexion de données vue du client

Politiques de sécurité du client :
Les clients n'ont normalement pas de "politiques" à proprement parler, ils s'appuient plutôt sur l'utilisateur pour définir leurs actions et, dans une certaine mesure, sont réactifs à la politique du serveur. Donc, un client aura besoin d'avoir des commandes qui vont permette à l'utilisateur de changer de façon dynamique le niveau de protection de la connexion de données ; cependant, il peut y avoir une "politique" générale qui tente d'abord toutes les commandes LIST et NLST sur une connexion "C" (et passe automatiquement à "P" si cela échoue). Dans ce cas, il va y avoir besoin qu'une commande de l'utilisateur soit disponible pour assurer qu'un certain transfert de données n'a pas été tenté sur une connexion de données non sûre.

Les clients ont aussi besoin de comprendre que le niveau du réglage de PROT n'est vérifié que pour un transfert de données particulier après que ce transfert ait été demandé. Donc, un refus du serveur d'accepter un transfert de données particulier ne devrait pas être lu par le client comme refus complet d'accepter de niveau de protection des données, car non seulement d'autres transferts de données peuvent être acceptables à ce niveau de protection, mais il est entièrement possible que le même transfert puisse être accepté au même niveau de protection un peu plus tard dans la session.

On notera que l'authentification TLS du client devrait être une authentification d'un utilisateur sur l'hôte client et non de l'hôte client lui-même.

11. Qui négocie quoi, où, et comment

11.1. Protège t-on quelque chose ?

Le client produit "AUTH TLS", le serveur accepte ou rejette. Si le serveur a besoin de AUTH, il refuse alors d'accepter certaines commandes jusqu'à ce qu'il obtienne une session qui réussisse à être protégée.

11.2 Quel niveau de protection utilise t-on sur la connexion de contrôle ?

Décidé entièrement par la négociation de suite de chiffrement TLS.

11.3 Protège t-on les connexions de données en général ?

Le client produit une commande PROT, que le serveur accepte ou rejette.

11.4 La protection est-elle exigée pour un transfert de données particulier ?

Un client aurait déjà produit une commande PROT si il exigeait que la connexion soit protégée.

Si un serveur a besoin d'une connexion protégée, il va alors répondre à la commande STOR/RETR/NLST/... par un code "522", indiquant que l'état actuel du niveau de protection de la connexion de données n'est pas suffisant pour l'instant pour ce transfert de données.

11.5 Quel niveau de protection est exigé pour un transfert de données particulier ?

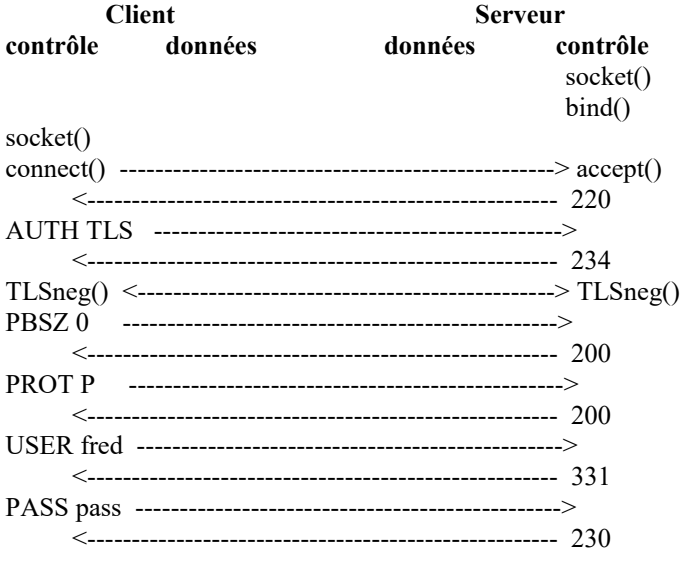
Décidé entièrement par la négociation de suite de chiffrement TLS.

Donc, pour plus de souplesse, on peut considérer qu'il est souhaitable que l'application FTP soit capable d'interagir avec la couche TLS sur laquelle elle se trouve pour définir et découvrir les suites de chiffrement TLS exactes qui sont à négocier, et de prendre les décisions qui en découlent.

12. Diagrammes temporels

Ces diagrammes temporels visent à aider à expliquer comment exactement s'intègrent la prise de contact TLS et la protection de session dans la logique existante du protocole FTP. Bien sûr, le protocole FTP lui-même n'est pas bien décrit par rapport à l'enchaînement des commandes et réponses dans la [RFC0959], de sorte que ceci se fonde partiellement sur l'observation empirique de mises en œuvre existantes largement répandues de client et de serveur.

12.1 Établissement d'une session protégée



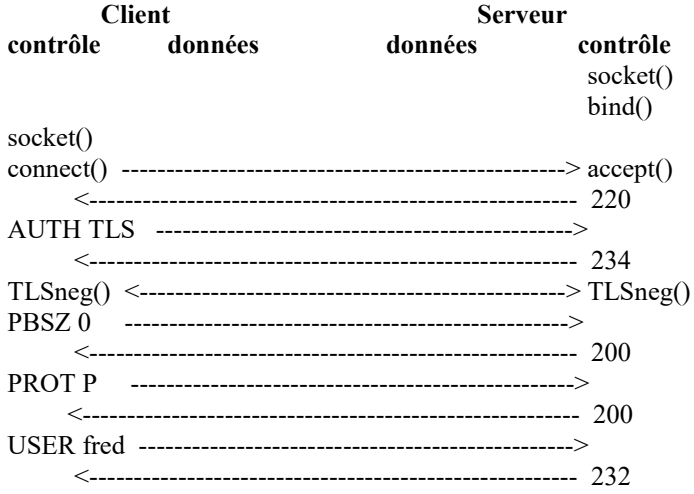
Note 1 : L'ordre de la paire PBSZ/PROT et de la paire USER/PASS (l'une par rapport à l'autre) n'est pas important (c'est-à-dire que USER/PASS peut arriver avant PBSZ/PROT, ou le serveur peut refuser de permettre une paire PBSZ/PROT jusqu'à ce que soit apparue la paire USER/PASS).

Note 2 : La commande PASS peut n'être pas requise du tout (si le paramètre USER et toute identité de client présentée fournit une authentification suffisante). Le serveur va indiquer cela en produisant une réponse "232" à la commande USER au lieu de "331", qui demande une PASS de la part du client (voir ci-dessous).

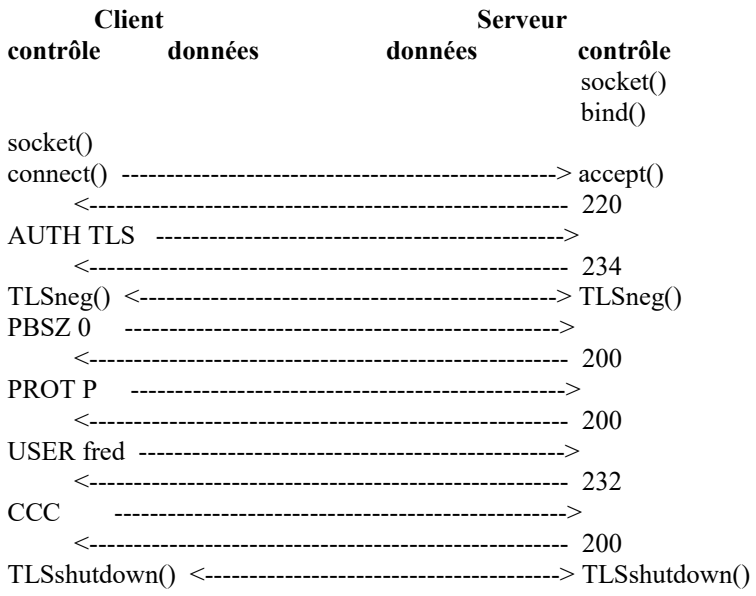
Note 3 : La commande AUTH peut n'être pas la première commande après la réception du message d'accueil "220".

12.2 Établissement d'une session protégée sans demande de mot de passe

(L'authentification TLS est suffisante).



12.3 Établissement d'une session protégée et suppression ensuite avec la commande CCC



- Le reste de la session de contrôle continue en clair avec les transferts de données protégés (à cause de PROT P).

Note : Ceci pose de sérieux problèmes de sécurité (voir la Section "Considérations sur la sécurité") mais peut être utile dans un scénario de pare-feu/NAT

12.4 Transfert de données standard sans protection

```
          Client                              Serveur
contrôle      données              données          contrôle
                socket()
                bind()
PORT w,x,y,z,a,b ---------------------------------------->
      <------------------------------------------------- 200
STOR file ---------------------------------------------->
                                     socket()
                                     bind()
      <------------------------------------------------- 150
            accept() <------------  connect()
            write()  ------------> read()
            close()  ------------> close()
      <------------------------------------------------- 226
```

12.5 Transfert de données sans protection à travers un pare-feu amical

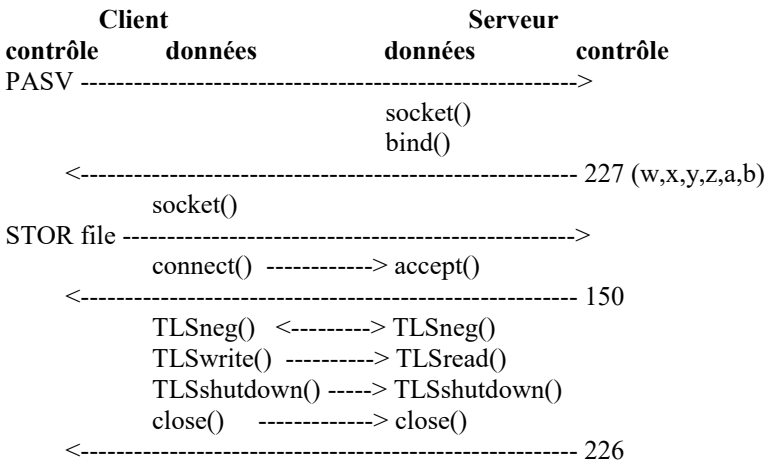
```
          Client                              Serveur
contrôle      données              données          contrôle
PASV --------------------------------------------------->
                                     socket()
                                     bind()
    <--------------------------------------------------- 227 (w,x,y,z,a,b)
            socket()
STOR file ---------------------------------------------->
            connect()  ----------> accept()
      <------------------------------------------------- 150
            write()    ----------> read()
            close()    ----------> close()
      <------------------------------------------------- 226
```

Note : Les développeurs devraient savoir que la fonction connect()/accept() est effectuée avant la réception de la réponse à la commande STOR. Cela diffère de la situation où un PORT qui n'accepte pas les pare-feu est utilisé avant le STOR, et le accept()/connect() est effectué avant que la réponse au STOR susmentionné ait été traitée.

12.6 Transfert de données standard avec protection

```
          Client                              Serveur
 contrôle     données              données          contrôle
                socket()
                bind()
PORT w,x,y,z,a,b ---------------------------------------->
      <------------------------------------------------- 200
STOR file ---------------------------------------------->
                                     socket()
                                     bind()
      <------------------------------------------------- 150
            accept()  <------------  connect()
            TLSneg()  <----------> TLSneg()
            TLSwrite() ----------> TLSread()
            TLSshutdown() -----> TLSshutdown()
            close()    ----------> close()
      <------------------------------------------------- 226
```

12.7 Transfert de données avec protection à travers un pare-feu amical



13. Discussion de la commande REIN

La commande REIN, définie dans la [RFC0959], permet à l'utilisateur de réinitialiser l'état de la session FTP. D'après la [RFC0959] :

REINITIALIZE (REIN)
Cette commande termine un USER, purgeant toutes les informations d'entrée/sortie et de comptabilité, sauf pour permettre d'achever tout transfert en cours. Tous les paramètres sont réinitialisés à leurs réglages par défaut et la connexion de contrôle est laissée ouverte. Ceci est identique à l'état dans lequel un utilisateur se trouve immédiatement après l'ouverture de la connexion de contrôle. On peut s'attendre à ce qu'une commande USER arrive à la suite.

Quand cette commande est traitée par le serveur, la ou les sessions TLS DOIVENT être purgées et les connexions de contrôle et de données reviennent à des communications en clair, non protégées. Il PEUT être acceptable d'utiliser des sessions TLS mises en antémémoire pour des connexions ultérieures, cependant, un serveur NE DOIT PAS le rendre obligatoire.

Si la commande REIN est utilisée pour supprimer une session TLS, la réponse à la commande REIN DOIT être envoyée dans une session protégée avant que la ou les sessions soient supprimées.

14. Discussion des commandes STAT et ABOR

Les commandes ABOR et STAT et l'utilisation des pointeurs TCP Urgent.

La [RFC0959] décrit l'utilisation des commandes Telnet (IP et DM) et le pointeur TCP Urgent pour indiquer la transmission de commandes sur le canal de contrôle durant l'exécution d'un transfert de données. FTP utilise les commandes Telnet Interrupt Process et Data Mark en conjonction avec Données urgentes pour précéder deux commandes : ABOR (interruption de transfert) et STAT (demande d'état).

Le pointeur Urgent était utilisé parce que, dans une mise en œuvre Unix, la réception d'un paquet TCP marqué Urgent résultait en l’exécution de l'opérateur d'interruption SIGURG. Cette dépendance à des opérateurs d'interruption était nécessaire sur les systèmes qui ne mettaient pas en œuvre select() ou ne prenaient pas en charge les trames multiples. TLS ne prend pas en charge la notion de données urgentes.

Quand TLS est mis en œuvre comme méthode de sécurité dans FTP, le serveur NE DEVRAIT PAS s’appuyer sur l'utilisation de SIGURG pour traiter les entrées sur le canal de contrôle durant les transferts de données. Le client DOIT envoyer toutes les données, y compris les commandes Telnet, à travers la session TLS.

15. Considérations sur la sécurité

Le présent document expose comment TLS peut être utilisé en conjonction avec la [RFC2228] pour fournir des mécanismes pour sécuriser les sessions FTP. Les discussions sur les raisons de la sécurité et les propriétés de sécurité sont contenues dans la [RFC2228] et ne sont pas répétées ici.

15.1 Vérification des jetons d'authentification

On suppose ici que les certificats X.509 seront utilisés pour l'authentification TLS. Si un autre jeton d'identité est utilisé (par exemple, des tickets Kerberos - voir la [RFC2712]), des considérations similaires, spécifiques du mécanisme, devront être appliquées.

15.1.1 Certificats de serveur

- Bien que ce soit entièrement une décision de la mise en œuvre, il est recommandé que les certificats utilisés pour l'authentification du serveur de la session TLS contiennent les informations d'identification du serveur d'une manière similaire à celle utilisée pour les serveurs http (voir la [RFC2818]).
- Il est fortement recommandé que le certificat utilisé pour l'authentification par le serveur des connexions de données soit le même que celui utilisé pour la connexion de contrôle correspondante. Si des certificats différents sont utilisés, il devrait y avoir un autre mécanisme que le client peut utiliser pour faire une vérification croisée des identités du serveur des connexions de données et de contrôle.
- Si des certificats de serveur ne sont pas utilisés, beaucoup des avantages pour la sécurité ne seront pas réalisés. Par exemple, dans un environnement Diffie-Hellman anonyme, il n'y a pas d'authentification de l'identité du serveur, de sorte qu'il y a peu de protection contre les attaques par interposition.

15.1.2 Certificats de client

- Décider quels certificats de client admettre et définir quels champs définissent quelles informations d'authentification est entièrement une question de mise en œuvre de serveur.
- Cependant, il est fortement recommandé que le certificat utilisé pour l'authentification du client des connexions de données soit le même certificat qu'utilisé pour la connexion de contrôle correspondante. Si des certificats différents doivent être utilisés, il devrait y avoir un autre mécanisme que le serveur puisse utiliser pour faire une vérification croisée des identités du client des connexions de données et de contrôle.
- Si des certificats de client ne sont pas utilisés, beaucoup des avantages de sécurité ne seront pas réalisés. Par exemple, il sera encore possible à un client malveillant de capturer une connexion de données.

15.2 Problèmes des considérations sur la sécurité de FTP de la [RFC2577]

15.2.1 Attaque par rebond

Une attaque par rebond devrait être plus difficile dans un environnement FTP sécurisé parce que :

- Le serveur FTP utilisé pour initier une connexion falsifiée sera toujours un "serveur" dans le contexte TLS. Donc, seuls les services qui agissent comme "clients" dans le contexte TLS pourront être vulnérables. Cela serait une façon contre-intuitive de mettre en œuvre TLS sur un service.
- Le serveur FTP va détecter que les accréditifs d'authentification pour la connexion de données ne sont pas les mêmes que pour la connexion de contrôle, donc les politiques de serveur pourraient être réglées à abandonner la connexion de données.
- Les utilisateurs authentiques ont peu de chances d'initier de telles attaques quand l'authentification est forte, et les utilisateurs malveillants ont moins de chances d'obtenir l'accès au serveur FTP si l'authentification n'est pas facile à subvertir (deviner le mot de passe, traçage du réseau, etc...)

15.2.2 Accès restreint

Le présent document propose un fort mécanisme pour résoudre le problème soulevé dans cette section.

15.2.3 Protection par mot de passe

Les solutions jumelles d'une forte authentification et de la confidentialité des données assurent que ceci n'est pas un problème quand TLS est utilisé pour protéger la session de contrôle.

15.2.4 Confidentialité

Le protocole TLS assure la confidentialité des données par le chiffrement. La défense de l'intimité (par exemple, l'accès aux fichiers téléchargés, les informations de profil d'utilisateur, etc...) sortent du domaine d'application du présent document (et vraisemblablement, de celui de la [RFC2577]).

15.2.5 Protection des noms d'utilisateur

Ce n'est pas un problème quand TLS est utilisé comme principal mécanisme d'authentification.

15.2.6. Vol d'accès

Cette spécification ne prévoit pas grand chose contre le déni de service ; cependant, une forte authentification sur la connexion de données empêchera des connexions non autorisées de récupérer ou soumettre des fichiers. Bien sûr, c'est seulement le cas lorsque une forte authentification du client est utilisée. Si les certificats de client ne sont pas utilisés, le vol d'accès par un client félon est toujours un problème. Si on utilise pas du tout d'authentification forte (par exemple, avec Diffie-Hellman anonyme) le problème du vol d'accès reste entier.

15.2.7 Problèmes de sécurité fondés sur le logiciel

Rien dans cette spécification n'affecte la discussion de ce paragraphe.

15.3 Problèmes de la commande CCC

L'utilisation de la commande CCC peut créer des problèmes de sécurité. Pour une description complète, voir la section "Canal de commande en clair (CCC)" de la [RFC2228]. Les clients ne devraient pas supposer qu'un serveur va permettre le traitement de la commande CCC.

Les mises en œuvre de serveur peuvent souhaiter refuser de traiter la commande CCC sur une session qui n'est pas passée par une forme quelconque d'authentification de client (par exemple, authentification de client TLS ou USER/PASS FTP). Cela peut empêcher des clients anonymes de demander de façon répétée AUTH TLS suivi par CCC pour ligoter les ressources sur le serveur.

16. Considérations relatives à l'IANA

{FTP-PORT} - L'accès alloué à la connexion de commande FTP est 21.

17. Autres paramètres

{TLS-PARM} - Le paramètre pour la commande AUTH pour indiquer que TLS est exigé. Pour demander le protocole TLS conformément au présent document, le client DOIT utiliser 'TLS'.

Pour conserver la rétro compatibilité avec les versions plus anciennes de ce document, le serveur DEVRAIT accepter "TLS-C" comme synonyme de "TLS".

Note : La [RFC2228] déclare que ces paramètres sont insensibles à la casse.

18. Adaptabilité et limites

Il n'y a pas de problèmes autres que ceux concernant la capacité du serveur de refuser d'avoir une négociation TLS complète pour chaque connexion de données, ce qui permet aux serveurs de conserver du débit tout en n'utilisant les cycles que quand c'est nécessaire.

19. Applicabilité

Ce mécanisme est d'application générale comme mécanisme de sécurisation du protocole FTP. Il est peu vraisemblable que des clients ou serveurs de FTP anonyme exigent un telle sécurité (bien que certains puissent apprécier les caractéristiques d'authentification sans la confidentialité).

20. Remerciements

- o à Netscape Communications Corporation pour le protocole SSL d'origine.
- o à Eric Young pour les bibliothèques SSLeay.
- o à l'Université de Californie, Berkeley pour les mises en œuvre originales de FTP et ftpd, sur lesquelles s'est nourrie la mise en œuvre initiale de ces extensions.
- o au groupe de travail CAT de l'IETF.
- o au groupe de travail TLS de l'IETF.
- o au groupe de travail FTPEXT de l'IETF.
- o à Jeff Altman pour la discussion sur ABOR et STAT.
- o au diverses personnes qui ont aidé l'auteur de ce document tout au long des étapes de préparation, à savoir Martin Carpenter, Eric Murray, Tim Hudson, et Volker Wiegand.

21. Références

21.1 Références normatives

[RFC0959] J. Postel et J. Reynolds, "Protocole de transfert de fichiers (FTP)", STD 9, octobre 1985. *(MàJ par* RFC7151)

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

[RFC2228] M. Horowitz, S. Lunt, "Extensions de sécurité pour FTP", octobre 1997. *(P.S.)*

[RFC2246] T. Dierks et C. Allen, "Protocole TLS version 1.0", janvier 1999. *(P.S. ; MàJ par* RFC7919)

[RFC2389] P. Hethmon, R. Elz, "Mécanisme de négociation de caractéristiques pour FTP", août 1998. *(P.S.)*

21.2 Références pour information

[RFC1579] S. Bellovin, "Pare-feu FTP facile", février 1994. *(Information)*

[RFC2222] J. Myers, "Authentification simple et couche de sécurité (SASL)", octobre 1997. (*Obsolète, voir* RFC4422, RFC4752) (*MàJ par* RFC2444) *(P.S.)*

[RFC2577] M. Allman, S. Ostermann, "Considérations sur la sécurité de FTP", mai 1999. *(Information)*

[RFC2712] A. Medvinsky, M. Hur, "Ajout des suites de chiffrement Kerberos à la sécurité de la couche transport (TLS)", octobre 1999. *(P.S.)*

[RFC2817] R. Khare, S. Lawrence, "Mise à niveau de TLS au sein de HTTP/1.1", mai 2000. *(P.S.)*

[RFC2818] E. Rescorla, "HTTP sur TLS", mai 2000. *(Information)*

[RFC3207] P. Hoffman, "Extension de service SMTP pour un SMTP sécurisé sur TLS", février 2002. *(P.S., MàJ par* RFC7817)

Contributeurs

Tim Hudson
RSA Data Security
Australia Pty Ltd
téléphone : +61 7 3227 4444
mél : tjh@rsasecurity.com.au

Volker Wiegand
SuSE Linux
mél : wiegand@suse.de

Martin Carpenter
Verisign Ltd
mél : mcarpenter@verisign.com

Eric Murray
Wave Systems Inc.
mél : ericm@lne.com

Adresse de l'auteur

Paul Ford-Hutchinson
IBM UK Ltd
PO Box 31
Birmingham Road
Warwick
United Kingdom

téléphone : +44 1926 462005
mél : rfc4217@ford-hutchinson.com

Déclaration complète de droits de reproduction

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