Groupe de travail Réseau
Request for Comments : 2228
RFC mise à jour : 959
Catégorie : En cours de normalisation

M. Horowitz, Cygnus Solutions
S. Lunt, Bellcore
octobre 1997
Traduction Claude Brière de L'Isle

Extensions de sécurité à FTP

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document définit des extensions à la spécification de FTP, STD 9, [RFC0959], "Protocole de transfert de fichiers (FTP)" (octobre 1985). Ces extensions fournissent une authentification forte, la protection de l'intégrité, et de la confidentialité à la fois sur les canaux de contrôle et de données avec l'introduction de nouvelles commandes facultatives, de réponses, et de codages de transfert de fichiers.
Les nouvelles commandes facultatives sont introduites dans la présente spécification :
AUTH (mécanisme d'authentification/sécurité),
ADAT (données d'authentification/sécurité),
PROT (niveau de protection du canal de données),
PBSZ *(Protection Buffer SiZe)* (taille de la mémoire tampon de protection),
CCC (canal de commande en clair),
MIC (commande d'intégrité protégée),
CONF (commande de confidentialité protégée),
ENC (commande de secret protégé).
Une nouvelle classe de types de réponses (6yz) est aussi introduite pour les réponses protégés.
La mise en œuvre d'aucune des commandes ci-dessus n'est exigée mais des interdépendances existent. Ces interdépendances sont documentées avec les commandes.
Noter que cette spécification est compatible avec le STD 9, [RFC0959].

Table des matières

1. Introduction

Le protocole de transfert de fichiers (FTP, *FileTransfer Protocol*) actuellement défini dans le STD 9, [RFC0959] et qui est en place sur l'Internet utilise les noms d'utilisateur et les mots de passe en clair pour authentifier les clients auprès des serveurs (via les commandes USER et PASS). Sauf pour des services comme les archives FTP "anonymes", cela représente un risque pour la sécurité car les mots de passe peuvent être volés suite à l'espionnage des réseaux locaux et de large zone. Cela aide d'une part les attaquants potentiels par l'exposition des mots de passe et/ou d'autre part limite l'accessibilité à des fichiers par des serveurs FTP qui ne peuvent pas ou ne veulent pas accepter ces risques inhérents à la sécurité.

À côté du problème de l'authentification sûre des utilisateurs, il y a aussi le problème de l'authentification des serveurs, de la protection des données sensibles et/ou de vérifier leur intégrité. Un agresseur peut être capable d'accéder à des données précieuses ou sensibles en surveillant simplement un réseau, ou peut être capable par des méthodes actives de supprimer ou modifier les données en cours de transfert de façon à corrompre leur intégrité. Un attaquant actif peut aussi initier des transferts parasites de fichiers à partir d'un site ou vers un site de son choix, et peut invoquer d'autres commandes sur le serveur. FTP n'a actuellement aucune disposition pour le chiffrement ou la vérification de l'authenticité des commandes, des réponses, ou des données transférées. Noter que ces services de sécurité sont précieux même sur l'accès anonyme de fichiers.

La pratique courante pour l'envoi sécurisé de fichiers est généralement soit :
1. via FTP de fichiers pré chiffrés avec des clés à distribution manuelle,
2 via un message électronique contenant un codage d'un fichier chiffré avec des clés qui sont distribuées manuellement,
3. via un message PEM,
4. via la commande rcp améliorée pour utiliser Kerberos.

Aucun de ces moyens ne pourrait être considéré comme un standard, même de fait, et aucun n'est vraiment interactif. Il existe un besoin de transférer en toute sécurité les fichiers en utilisant FTP d'une manière sécurisée qui soit prise en charge de façon cohérente au sein du protocole FTP et qui tire parti de l'infrastructure et de la technologie existante. Les extensions sont nécessaires à la spécification FTP si ces services de sécurité doivent être introduits dans le protocole de façon interopérable.

Bien que la connexion de contrôle FTP suive le protocole Telnet, et que Telnet ait défini une option d'authentification et de chiffrement [RFC2941], la [RFC1123] interdit explicitement l'utilisation de la négociation d'option Telnet sur la connexion de contrôle (autre que Synch et IP).

Aussi, l'option Telnet d'authentification et de chiffrement ne fournit pas la protection de l'intégrité seule (sans la confidentialité) et ne traite pas de la protection du canal des données.

2. Vue générale de la sécurité sur FTP

Au plus haut niveau, les extensions de sécurité à FTP cherchent à donner un mécanisme abstrait pour authentifier et/ou autoriser les connexions, et des commandes de protection de l'intégrité et/ou la confidentialité, les transferts des réponses, et des données.

Dans le contexte de la sécurité de FTP, l'authentification est l'établissement de l'identité d'un client et/ou d'un serveur d'une façon sûre, usuellement en utilisant des techniques cryptographiques. Le protocole FTP de base ne possède pas de concept d'authentification.

L'autorisation est le processus de validation d'un utilisateur pour la connexion. Le processus d'autorisation de base implique les commandes USER, PASS, et ACCT. Avec les extensions de sécurité à FTP, l'authentification établie en utilisant un mécanisme de sécurité peut aussi être utilisée pour prendre la décision d'autorisation.

Sans les extensions de sécurité, l'authentification du client, selon l'acception usuelle de ce terme, ne se produit jamais. L'autorisation FTP est accomplie avec un mot de passe, passé en clair sur le réseau comme argument de la commande PASS. Le possesseur de ce mot de passe est supposé être autorisé à transférer des fichiers en tant qu'utilisateur désigné dans la commande USER, mais l'identité du client n'est jamais établie de façon sûre.

Une interaction de sécurité FTP commence par un client qui dit au serveur quel mécanisme de sécurité il veut utiliser avec la commande AUTH. Le serveur va accepter ce mécanisme, le rejeter, ou , dans le cas d'un serveur qui ne met pas en œuvre les extensions de sécurité, rejeter complètement la commande. Le client peut essayer plusieurs mécanismes de sécurité jusqu'à ce qu'il en demande un que le serveur accepte. Cela permet à une forme rudimentaire de négociation d'avoir lieu. (Si on désire une négociation plus complexe, cela peut être mis en œuvre comme un mécanisme de sécurité.) La réponse du

serveur va indiquer si le client doit répondre par des données supplémentaires à interpréter par le mécanisme de sécurité. Si aucune n'est nécessaire, cela va normalement signifier que le mécanisme est un de ceux où le mot de passe (spécifié par la commande PASS) est à interpréter différemment, comme avec un jeton ou un système à mot de passe à utilisation unique.

Si le serveur exige des informations de sécurité supplémentaires, le client et le serveur vont alors entrer dans un échange de données de sécurité. Le client va envoyer une commande ADAT contenant le premier bloc des données de sécurité. La réponse du serveur va indiquer si l'échange de données est terminé, si il y avait une erreur, ou si plus de données sont nécessaires. La réponse du serveur peut facultativement contenir des données de sécurité que le client devra interpréter. Si plus de données sont nécessaires, le client va envoyer une autre commande ADAT contenant le bloc de données suivant, et attendre la réponse su serveur. Cet échange peut continuer aussi longtemps que nécessaire. Une fois que cet échange est terminé, le client et le serveur ont établi une association de sécurité. Cette association de sécurité peut inclure l'authentification (du client, du serveur, ou mutuelle) et des informations de clés pour l'intégrité et/ou la confidentialité, selon le mécanisme utilisé.

Le terme "données de sécurité" est ici choisi avec soin. L'objet de l'échange de données de sécurité est d'établir une association de sécurité, qui peut en fait ne pas inclure du tout d'authentification, entre le client et le serveur, comme décrit ci-dessus. Par exemple, un échange Diffie-Hellman établit une clé secrète, mais aucune authentification n'a lieu. Si un serveur FTP a une paire de clés RSA mais pas le client, celui-ci peut authentifier le serveur, mais le serveur ne peut pas authentifier le client.

Une fois qu'est établie une association de sécurité, l'authentification qui fait partie de cette association peut être utilisée à la place de, ou en plus, de l'échange standard de nom d'utilisateur/mot de passe pour autoriser un usager à se connecter au serveur. Un nom d'utilisateur spécifié par la commande USER est toujours exigé pour spécifier l'identité à utiliser sur le serveur.

Afin d'empêcher un agresseur d'insérer ou supprimer des commandes dans le flux de contrôle, si l'association de sécurité prend en charge la protection de l'intégrité, le serveur et le client doivent alors utiliser la protection de l'intégrité sur le flux de contrôle, à moins de transmettre d'abord une commande CCC pour désactiver cette exigence. La protection de l'intégrité est effectuée avec les commandes MIC et ENC, et les codes de réponse 63z. La commande CCC et sa réponse doivent être transmises avec la protection d'intégrité. Les commandes et réponses ne peuvent être transmises sans la protection d'intégrité (c'est-à-dire, en clair ou avec seulement la confidentialité) que si aucune association de sécurité n'est établie, si l'association de sécurité négociée ne prend pas en charge l'intégrité, ou si la commande CCC a réussi.

Une fois que le client et le serveur ont négocié avec la commande PBSZ une taille acceptable de mémoire tampon pour encapsuler les données protégées sur le canal de données, le mécanisme de sécurité peut aussi être utilisé pour protéger les transferts de canal de données.

La politique n'est pas spécifiée dans le présent document. En particulier, les mises en œuvre de client et de serveur peuvent choisir de mettre en œuvre des restrictions sur les opérations qui peuvent être effectuées selon l'association de sécurité existante. Par exemple, un serveur peut exiger qu'un client autorise via un mécanisme de sécurité plutôt qu'en utilisant un mot de passe, exiger que le client fournisse un mot de passe à utilisation unique à partir d'un jeton, exiger au moins la protection de l'intégrité sur le canal de commande, ou exiger que certains fichiers ne soient transmis que chiffrés. Un client FTP anonyme peut refuser de faire des transferts de fichiers sans protection de l'intégrité afin de s'assurer de la validité des fichiers téléchargés.

Aucun ensemble particulier de fonctionnalités n'est exigé, sauf de façon subordonnée, comme décrit à la section suivante. Cela signifie que ni l'authentification, ni l'intégrité, ni la confidentialité ne sont exigées d'une mise en œuvre, bien qu'un mécanisme qui ne ferait rien de tout cela ne servirait pas à grand chose. Par exemple, il est acceptable pour un mécanisme de ne mettre en œuvre que la protection de l'intégrité, l'authentification dans une seule direction et/ou le chiffrement, le chiffrement sans aucune authentification ou protection de l'intégrité, ou tout autre sous-ensemble de fonctionnalités si des considérations politiques ou techniques le rendent souhaitable. Bien sûr, un des homologues peut exiger au titre de sa politique une protection plus forte que ce que l'autre est capable de fournir, empêchant une interopérabilité parfaite.

3. Nouvelles commandes FTP

Les commandes suivantes sont facultatives, mais dépendent les unes des autres. Ce sont des extensions aux commandes de contrôle d'accès FTP.

Les codes de réponse documentés ici sont généralement décrits comme recommandés, plutôt qu'exigés. L'intention est que les codes de réponse qui décrivent la gamme complète des modes de réussite et d'échec existent, mais qu'il soit permis aux serveurs de limiter les informations présentées au client. Par exemple, un serveur peut mettre en œuvre un mécanisme de

sécurité particulier, mais avoir une restriction de politique à l'égard de son utilisation. Le serveur devrait répondre par un code de réponse de 534 dans ce cas, mais peut réponde avec un code de réponse 504 si il ne souhaite pas divulguer que le mécanisme refusé est pris en charge. Si le serveur choisit bien d'utiliser un code de réponse différent de celui qui est recommandé, il devrait essayer d'utiliser un code de réponse qui n'en diffère que par le dernier chiffre. Dans tous les cas, le serveur doit utiliser un code de réponse qui est documenté comme retournable par la commande reçue, et ce code de réponse doit commencer par le même chiffre que le code de réponse recommandé pour cette situation.

Mécanisme Authentification/Sécurité (AUTH)

Le champ argument est une chaîne Telnet qui identifie un mécanisme pris en charge. Cette chaîne est insensible à la casse. Les valeurs doivent être enregistrées auprès de l'IANA, excepté les valeurs qui commencent par "X-" qui sont réservées pour une utilisation locale.

Si le serveur ne reconnaît pas la commande AUTH, il doit répondre avec le code de réponse 500. Ceci est destiné à englober la large base déployée de serveurs FTP sans capacité de sécurité, qui vont répondre avec le code 500 à toute commande non reconnue. Si le serveur reconnaît bien la commande AUTH mais ne met pas en œuvre les extensions de sécurité, il devrait répondre avec le code de réponse 502.

Si le serveur ne comprend pas le mécanisme de sécurité désigné, il devrait répondre par le code 504.

Si le serveur ne veut pas accepter le mécanisme de sécurité désigné, il devrait répondre avec le code 534.

Si le serveur n'est pas capable d'accepter le mécanisme de sécurité désigné, comme si une ressource demandée est indisponible, il devrait répondre avec le code 431.

Si le serveur veut accepter le mécanisme de sécurité désigné, mais exige des données de sécurité, il doit répondre avec le code 334.

Si le serveur veut accepter le mécanisme de sécurité désigné, et s'il n'exige pas de données de sécurité, il doit répondre avec le code 234.

Si le serveur répond avec un code 334, il peut inclure des données de sécurité comme décrit à la section suivante.

Certains serveurs vont permettre que la commande AUTH soit répétée afin d'établir une nouvelle authentification. La commande AUTH, si elle est acceptée, retire tout état associé aux commandes de sécurité FTP antérieures. Le serveur doit aussi exiger que l'utilisateur renouvelle l'autorisation (c'est-à-dire, reproduise certaines ou toutes les commandes USER, PASS, et ACCT) dans ce cas (voir à la section 4 l'explication de "autoriser" dans ce contexte).

Données d'authentification/sécurité (ADAT)

Le champ argument est une chaîne Telnet qui représente des données de sécurité codées en base 64 (voir à la Section 9, "Codage en base 64"). Si un code de réponse indiquant le succès est retourné, le serveur peut aussi utiliser une chaîne de la forme "ADAT=base64data" comme partie texte de la réponse si il souhaite renvoyer des données de sécurité au client.

Dans les deux cas, les données sont spécifique du mécanisme de sécurité spécifié par la précédente commande AUTH. La commande ADAT, et les réponses associées, permettent au client et au serveur de suivre n'importe quel protocole de sécurité. L'échange de données de sécurité doit inclure suffisamment d'informations pour que les deux homologues sachent quelles caractéristiques facultatives sont disponibles. Par exemple, si le client ne prend pas en charge le chiffrement des données, le serveur doit en être informé, afin qu'il sache qu'il ne faut pas envoyer de réponses chiffrées sur le canal de commandes. Il est vivement recommandé que le mécanisme de sécurité fournisse le séquençage sur le canal de commandes, pour s'assurer que des commandes ne sont pas supprimées, réarrangées ou répétées.

La commande ADAT doit être précédée par une commande AUTH réussie, et ne peut pas être produite une fois qu'un échange de données de sécurité s'est terminé (avec ou sans succès) sauf si il est précédé par une commande AUTH de restaurer l'état de sécurité.

Si le serveur n'a pas encore reçu une commande AUTH, ou si un échange de données précédent s'est achevé, mais l'état de sécurité n'a pas encore été restauré avec une commande AUTH, il devrait répondre par un code 503.

Si le serveur ne peut pas décoder l'argument en base 64, il devrait répondre par le code 501.

Si le serveur rejette les données de sécurité (si une somme de contrôle échoue, par exemple) il devrait répondre par un code 535.

Si le serveur accepte les données de sécurité, et exige des données supplémentaires, il devrait répondre par un code 335.

Si le serveur accepte les données de sécurité, mais n'exige pas de données supplémentaires (c'est-à-dire, si l'échange de données de sécurité s'est terminé avec succès) il doit répondre par un code 235.

Si le serveur répond par un code 235 ou 335, il peut inclure des données de sécurité dans la partie texte de la réponse, comme spécifié ci-dessus.

Si la commande ADAT retourne une erreur, l'échange de données de sécurité va échouer, et le client doit restaurer son état de sécurité interne. Si le client perd la synchronisation avec le serveur (par exemple, le serveur envoie un code de réponse 234 à une commande AUTH, mais le client a d'autres données à transmettre) le client doit alors restaurer l'état de sécurité du serveur.

Taille de la mémoire tampon de protection (PBSZ)

L'argument est un entier décimal qui représente la taille maximum, en octets, des blocs de données codés à envoyer ou recevoir durant le transfert de fichier. Ce nombre ne devra pas être supérieur à ce qui peut être représenté par un entier non signé de 32 bits.

Cette commande permet au client et serveur FTP de négocier une taille maximum de mémoire tampon protégée pour la connexion. Il n'y a pas de taille par défaut ; le client doit produire une commande PBSZ avant de pouvoir produire la première commande PROT.

La commande PBSZ doit être précédée par un échange réussi de données de sécurité.

Si le serveur ne peut pas analyser l'argument, ou si il ne tient pas dans 32 bits, il devrait répondre par un code 501.

Si le serveur n'a pas terminé un échange de données de sécurité avec le client, il devrait répondre par un code 503.

Autrement, le serveur doit répondre par un code 200. Si la taille fournie par le client est trop grande pour le serveur, il doit utiliser une chaîne de la forme "PBSZ=nombre" dans la partie texte de la réponse pour indiquer une plus petite taille de mémoire tampon. Le client et le serveur doivent utiliser la plus petite des deux tailles de mémoire tampon si les deux tailles de mémoire tampon sont spécifiées.

Niveau de protection du canal de données (PROT)

L'argument est un seul code de caractère Telnet qui spécifie le niveau de protection du canal des données.

Cette commande indique au serveur quel type de protection du canal des données le client et le serveur vont utiliser. Les codes suivants sont alloués :

C - Clair
S – Sûr
E - Confidentiel
P - Privé

Le niveau de protection par défaut si aucun autre niveau n'est spécifié est Clair. Le niveau de protection Clair indique que le canal des données va porter les données brutes du transfert de fichier, sans aucune application de sécurité. Le niveau de protection Sûr indique que les données seront protégées en intégrité. Le niveau de protection Confidentiel indique que la confidentialité des données sera protégée. Le niveau de protection Privé indique que l'intégrité et la confidentialité des données seront protégées.

Il est raisonnable qu'un mécanisme de sécurité ne fournisse pas tous les niveaux de protection du canal des données. Il est aussi raisonnable qu'un mécanisme fournisse plus de protection que requis à un certain niveau (par exemple, un mécanisme pourrait fournir la protection de Confidentiel, mais inclure la protection de l'intégrité dans ce codage du fait de l'API ou d'autres considérations).

La commande PROT doit être précédée par une négociation réussie de la taille de la mémoire tampon de protection.

Si le serveur ne comprend pas le niveau de protection spécifié, il devrait répondre avec le code 504.

Si le mécanisme de sécurité actuel ne prend pas en charge le niveau de protection spécifié, le serveur devrait répondre par le code 536.

Si le serveur n'a pas achevé la négociation de la taille de la mémoire tampon de protection avec le client, il devrait répondre par le code 503.

La commande PROT sera rejetée et le serveur devrait répondre le code 503 si une commande PBSZ n'a pas été produite précédemment.

Si le serveur ne veut pas accepter le niveau de protection spécifié, il devrait répondre par le code 534.

Si le serveur n'est pas capable d'accepter le niveau de protection spécifié, si par exemple une ressource nécessaire est indisponible, il devrait répondre le code 431.

Autrement, le serveur doit répondre par un code 200 pour indiquer que le niveau de protection spécifié est accepté.

Supprimer le canal de commande (CCC, *Clear Command Channel*)

Cette commande ne prend pas d'argument.

Il est souhaitable dans certains environnements d'utiliser un mécanisme de sécurité pour authentifier et/ou autoriser le client et le serveur, mais pas pour effectuer une vérification d'intégrité sur les commandes suivantes. Cela pourrait être utilisé dans un environnement où la sécurité IP est en place, assurant que les hôtes sont authentifiés et que les flux TCP ne peuvent pas être altérés, mais où on désire que l'utilisateur soit authentifié.

Si des commandes non protégées sont admises sur une connexion, un agresseur pourrait alors insérer une commande dans le flux de contrôle, et le serveur n'aurait aucun moyen de savoir qu'elle était invalide. Afin d'empêcher de telles attaques, une fois qu'un échange de données de sécurité s'est achevé avec succès, si le mécanisme de sécurité prend en charge l'intégrité, celle ci doit alors être utilisée (via la commande MIC ou ENC, et la réponse 631 ou 632) jusqu'à ce que la commande CCC soit produite pour activer les messages non protégés en intégrité sur le canal de contrôle. La commande CCC elle-même doit être protégée en intégrité.

Une fois la commande CCC achevée avec succès, si une commande n'est pas protégée, la réponse à cette commande doit aussi n'être pas protégée. Cela est destiné à prendre en charge l'interopérabilité avec les clients qui ne prennent pas en charge la protection une fois que la commande CCC a été produite.

Cette commande doit être précédée d'un échange réussi de données de sécurité.

Si la commande n'est pas protégée en intégrité, le serveur doit répondre par un code 533.

Si le serveur ne veut pas supprimer l'exigence d'intégrité, il devrait répondre avec un code 534.

Autrement, le serveur doit répondre avec un code 200 pour indiquer que les commandes et réponses non protégées peuvent maintenant être utilisées sur le canal de commandes.

Commandes Intégrité protégée (MIC) Confidentialité protégée (CONF) et Secret protégé (ENC)

Le champ d'argument de MIC est une chaîne Telnet qui consiste en un message "sûr" codé en base 64 produit par une procédure d'intégrité de message spécifique du mécanisme de sécurité. Le champ d'argument de CONF est une chaîne Telnet consistant en un message "confidentiel" codé en base 64 produit par une procédure de confidentialité de message spécifique du mécanisme de sécurité. Le champ d'argument de ENC est une chaîne Telnet consistant en un message "secret" codé en base 64 produit par une procédure d'intégrité et de confidentialité de message spécifique du mécanisme de sécurité.

Le serveur va décoder et/ou vérifier le message codé.

Cette commande doit être précédée par un échange réussi de données de sécurité.

Un serveur peut exiger que la première commande après un échange réussi de données de sécurité soit CCC, et de ne pas du tout mettre en œuvre les commandes de protection. Dans ce cas, le serveur devrait répondre avec un code 502.

Si le serveur ne peut pas décoder l'argument en base 64, il devrait répondre avec un code 501.

Si le serveur n'a pas terminé un échange de données de sécurité avec le client, il devrait répondre avec un code 503.

Si le serveur a terminé un échange de données de sécurité avec le client en utilisant un mécanisme qui prend en charge l'intégrité, et exige une commande CCC du fait de limitations de la politique ou de la mise en œuvre, il devrait répondre avec un code 503.

Si le serveur rejette la commande parce qu'elle n'est pas prise en charge par le mécanisme actuel de sécurité, le serveur devrait répondre avec le code 537.

Si le serveur rejette la commande (sur échec d'une somme de contrôle par exemple) il devrait répondre par un code 535.

Si le serveur ne veut pas accepter la commande (par exemple, si le secret est exigé par la politique, ou si une commande CONF est reçue avant une commande CCC) il devrait répondre avec le code 533.

Autrement, la commande sera interprétée comme une commande FTP. Il n'est pas nécessaire d'inclure un code de fin de ligne, mais s'il en est un d'inclus, il doit être un code de fin de ligne Telnet, et non un code de fin de ligne local.

Le serveur peut exiger, dans certaines circonstances ou dans toutes, que toutes les commandes soient protégées. Dans ce cas, il devrait faire une réponse 533 aux commandes autres que MIC, CONF, et ENC.

4. Autorisation de connexion

L'échange de données de sécurité peut, entre autres choses, établir de façon sûre l'identité du client auprès du serveur. Cette identité peut être utilisée comme entrée dans le processus d'autorisation de connexion.

En réponse aux commandes de connexion FTP (AUTH, PASS, ACCT) le serveur peut choisir de changer la séquence des commandes et réponses spécifiée dans la [RFC0959] comme suit. Il y a aussi de nouvelles réponses disponibles.

Si le serveur veut permettre à l'utilisateur désigné par la commande USER de se connecter sur la base de l'identité établie par l'échange de données de sécurité, il devrait répondre avec le code 232.

Si le mécanisme de sécurité exige un mot de passe à mise au défi/réponse, il devrait répondre à la commande USER avec le code 336. La partie texte de la réponse devrait contenir la mise au défi. Le client doit dans ce cas afficher la mise au défi à l'utilisateur avant d'inviter à produire le mot de passe. Ceci est particulièrement pertinent pour des clients plus sophistiqués ou des interfaces d'utilisateur graphiques qui fournissent des boîtes de dialogue ou d'autres entrées modales. Ces clients devraient veiller à ne pas inviter à produire le mot de passe avant que le nom d'utilisateur ait été envoyé au serveur, au cas où l'utilisateur aurait besoin de la mise au défi dans la réponse 336 pour construire un mot de passe valide.

5. Nouvelles réponses FTP

Les nouveaux codes de réponse sont divisés en deux classes. La première classe est faite de nouvelles réponses rendues nécessaires par les nouvelles commandes de sécurité FTP. La seconde classe est un nouveau type de réponse pour indiquer des réponses protégées.

5.1 Nouveaux codes de réponse individuels

232 Utilisateur connecté, autorisé par l'échange de données de sécurité.
234 Échange de données de sécurité terminé.
235 [ADAT=base64data] ; Cette réponse indique que l'échange de données de sécurité s'est terminé avec succès. Les crochets ne sont pas à inclure dans la réponse, mais indiquent que ces données de sécurité sont facultatives dans la réponse.
334 [ADAT=base64data] ; Cette réponse indique que le mécanisme de sécurité demandé est correct et comporte des données de sécurité à utiliser par le client pour construire la prochaine commande. Les crochets ne sont pas à inclure dans la réponse, mais indiquent que ces données de sécurité sont facultatives dans la réponse.
335 [ADAT=base64data] ; Cette réponse indique que les données de sécurité sont acceptables, et que d'autres sont requises pour terminer l'échange de données de sécurité. Les crochets ne sont pas à inclure dans la réponse, mais indiquent que ces données de sécurité sont facultatives dans la réponse.
336 Le nom d'utilisateur est correct, il faut le mot de passe. Le défi est "...." ; La représentation exacte du défi devrait être choisie par le mécanisme pour être compréhensible par l'utilisateur humain du système.
431 Une ressource indisponible est nécessaire pour traiter la sécurité.

533 Le niveau de protection de la commande est refusé pour des raisons de politique.
534 Demande refusée pour des raisons de politique.
535 Échec d'une vérification de sécurité (hachage, séquence, etc.).
536 Le niveau de PROT demandé n'est pas pris en charge par le mécanisme.
537 Le niveau de protection de la commande n'est pas accepté par le mécanisme de sécurité.

5.2 Réponses protégées

Un nouveau type de réponse est introduit :

6yz Réponse protégée

Il y a trois codes de réponse de ce type. Le premier, le code 631, indique une réponse protégée en intégrité. Le second, le code 632, indique une réponse protégée en confidentialité et en intégrité. Le troisième, le code 633, indique une réponse protégée en confidentialité.

La partie texte d'une réponse 631 est une chaîne Telnet consistant en un message "sûr" codé en base 64 et produit par une procédure d'intégrité de message spécifique du mécanisme de sécurité. La partie texte d'une réponse 632 est une chaîne Telnet consistant en un message "secret" codé en base 64 produit par une procédure de confidentialité et d'intégrité de message spécifique du mécanisme de sécurité. La partie texte d'une réponse 633 est une chaîne Telnet consistant en un message "confidentiel" codé en base 64 produit par une procédure de confidentialité de message spécifique du mécanisme de sécurité.

Le client va décoder et vérifier la réponse codée. La façon de traiter les échecs du décodage ou de la vérification des réponses est spécifique de la mise en œuvre. Un code de fin de ligne n'est pas nécessairement inclus, mais s'il en est un, il doit être un code Telnet de fin de ligne, et non un code local de fin de ligne.

Une réponse protégée ne peut être envoyée que si un échange de données de sécurité a réussi.

La réponse 63z peut être sur plusieurs lignes. Dans ce cas, la réponse en clair doit être coupée en un certain nombre de fragments. Chaque fragment doit être protégé, puis codé en base 64 dans l'ordre sur une ligne distincte de la réponse multi ligne. Il n'est pas nécessaire qu'il y ait de correspondance entre les coupures de ligne de la réponse en clair et de la réponse codée. Les codes de fin de ligne Telnet doivent apparaître dans le texte en clair de la réponse codée, sauf pour le code de la fin de ligne finale, ce qui est facultatif.

La réponse multi-ligne doit être formatée de façon plus stricte que dans la spécification de continuation de la RFC 959. En particulier, chaque ligne avant la dernière doit être formée par le code de réponse, suivi immédiatement par un trait d'union, suivi par un fragment codé en base 64 de la réponse.

Par exemple, si la réponse en clair est

```
123-Première ligne
   Seconde ligne
   234 Une ligne commençant par les nombres
123 La dernière ligne
```

puis la réponse protégée résultante pourrait être une des suivantes (le premier exemple n'a une coupure de ligne que pour tenir dans les marges) :

```
631 base64(protect("123-Première ligne\r\nSeconde ligne\r\n  234 Une ligne
631-base64(protect("123-Première ligne\r\n"))
631-base64(protect("Seconde ligne\r\n"))
631-base64(protect("  234 Une ligne commençant par les nombres\r\n"))
631 base64(protect("123 La dernière ligne"))
631-base64(protect("123-Première ligne\r\nSeconde ligne\r\n  234 Une ligne c"))
631 base64(protect("mmençant par les nombres\r\n123 La dernière ligne\r\n"))
```

6. Encapsulation du canal des données

Lorsque les transferts de données sont protégés entre le client et le serveur (dans l'une ou l'autre direction) certaines transformations et encapsulations doivent être effectuées de sorte que le receveur puisse décoder correctement le fichier transmis.

L'envoyeur doit appliquer tous les services de protection après que les transformations associées au type de représentation, à la structure du fichier, et au mode de transfert ont été effectuées. Les données envoyées sur le canal des données sont, pour les besoins de la protection, à traiter comme un flux d'octets.

Lorsque on effectue un transfert de données d'une manière authentifiée, les vérifications d'authentification sont effectuées sur les blocs individuels du fichier, plutôt que sur l'ensemble du fichier. Par conséquent, il est possible qu'une attaque par insertion introduise des blocs dans le flux des données (c'est-à-dire, des répétitions) qui s'authentifient correctement, mais résultent en un fichier corrompu qui n'est pas détecté par le receveur. Pour se garder contre de telles attaques, le mécanisme de sécurité spécifique employé devrait inclure des mécanismes qui protègent contre ces attaques. Ceci est fait par de nombreux mécanismes d'API GSS utilisables avec cette spécification à l'Appendice I, et par le mécanisme Kerberos de l'Appendice II.

L'envoyeur doit prendre le flux d'octets en entrée et le couper en blocs de telle sorte que chaque bloc, lorsque il est codé en utilisant une procédure spécifique du mécanisme de sécurité, ne soit pas plus grand que la taille de mémoire tampon négociée par le client avec la commande PBSZ. Chaque bloc doit être codé, puis transmis avec la longueur du bloc codé ajoutée en tête comme un entier de quatre octets non signé, l'octet de poids fort en premier.

Lorsque la fin du fichier est atteinte, l'envoyeur doit coder un bloc de zéro octet, et envoyer ce bloc final au receveur avant de clore la connexion des données.

Le receveur va lire les quatre octets de longueur, lire un bloc de données de ce nombre d'octets, puis décoder et vérifier ce bloc avec une procédure spécifique du mécanisme de sécurité. Cela doit être répété jusqu'à ce que soit reçu un bloc codant une mémoire tampon de zéro octet. Cela indique la fin du flux d'octets codé.

Toutes les transformations associées au type de représentation, à la structure du fichier, et au mode de transfert sont à effectuer par le receveur sur le flux d'octets résultant du processus ci-dessus.

Lorsque on utilisé le mode de transfert de bloc, la taille de la mémoire tampon de l'envoyeur (du texte en clair) est indépendante de la taille du bloc.

Le serveur va répondre 534 à une commande STOR, STOU, RETR, LIST, NLST, ou APPE si le niveau de protection actuel n'est pas celui dicté par les exigences de sécurité du serveur pour ce transfert de fichier particulier.

Si un service de protection de données échoue à un moment quelconque durant le transfert de données à l'extrémité serveur (y compris une tentative d'envoi d'une taille de mémoire tampon supérieure au maximum négocié) le serveur enverra une réponse 535 à la commande de transfert de données (STOR, STOU, RETR, LIST, NLST, ou APPE).

7. Considérations de politique potentielles

Bien qu'il n'y ait pas de restriction sur la politique du client ni du serveur, il y a quelques recommandations qu'une mise en œuvre devrait prendre en considération.

- Une fois que commence un échange de données de sécurité, un serveur devrait exiger que toutes les commandes soient protégées (en intégrité et/ou confidentialité) et il devrait protéger toutes les réponses. Les réponses devraient utiliser le même niveau de protection que la commande qui les a produites. Cela inclut les réponses qui indiquent une défaillance des commandes MIC, CONF, et ENC. En particulier, il est tout à fait sensé d'exiger que AUTH et ADAT soient protégées ; il est sensé et utile d'exiger que PROT et PBSZ soient protégées. En particulier, l'utilisation de CCC n'est pas recommandée, mais elle est définie dans l'intérêt de l'interopérabilité entre les mises en œuvre qui pourraient souhaiter une telle fonctionnalité.

- Un client devrait chiffrer la commande PASS chaque fois que possible. Il est raisonnable pour le serveur de refuser d'accepter une commande PASS non chiffrée si le serveur dit que le chiffrement est disponible.

- Bien que la mise en œuvre d'aucune commande de sécurité ne soit exigée, il est recommandé qu'une mise en œuvre fournisse toutes les commandes qui peuvent être appliquées, en fonction des mécanismes pris en charge et des considérations de politique du site (contrôles des exportations, par exemple).

8. Spécifications déclaratives

Les paragraphes qui suivent sont sur le modèle des paragraphes 5.3 et 5.4 de la [RFC0959], qui décrivent les mêmes informations, sauf pour les commandes et réponses FTP standard.

8.1 Commandes et arguments de sécurité FTP

AUTH <SP> <nom-du-mécanisme> <CRLF>
ADAT <SP> <base64data> <CRLF>
PROT <SP> <prot-code> <CRLF>
PBSZ <SP> <entier-décimal> <CRLF>
MIC <SP> <base64data> <CRLF>
CONF <SP> <base64data> <CRLF>
ENC <SP> <base64data> <CRLF>

<nom-du-mécanisme> ::= <chaîne>
<base64data> ::= <chaîne> ; doit être formatée comme décrit à la section 9
<prot-code> ::= C | S | E | P
<entier-décimal> ::= tout entier décimal de 1 à (2^32)-1

8.2 Séquences de commande-réponse

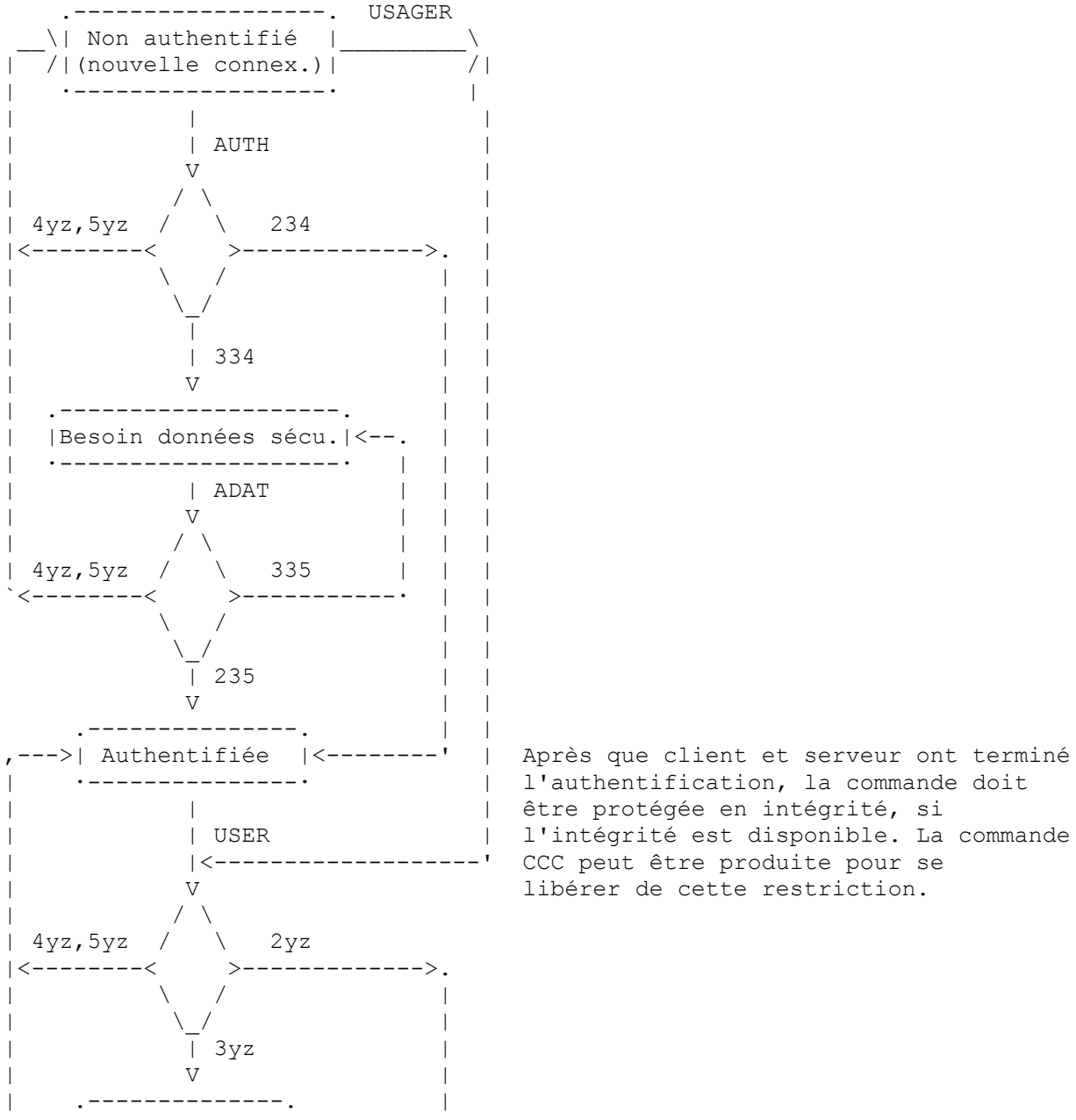
Établissement d'association de sécurité
AUTH
234
334
502, 504, 534, 431
500, 501, 421
ADAT
235
335
503, 501, 535
500, 501, 421
Commandes de négociation de protection des données
PBSZ
200
503
500, 501, 421, 530
PROT
200
504, 536, 503, 534, 431
500, 501, 421, 530
Commandes de protection du canal de commande
MIC
535, 533
500, 501, 421
CONF
535, 533
500, 501, 421
ENC
535, 533
500, 501, 421
Commandes de connexion à sécurité améliorée (seules les nouvelles réponses sont énumérées)
USER
232
336
Commandes de canal de données (seules les nouvelles réponses sont énumérées)
STOR
534, 535
STOU

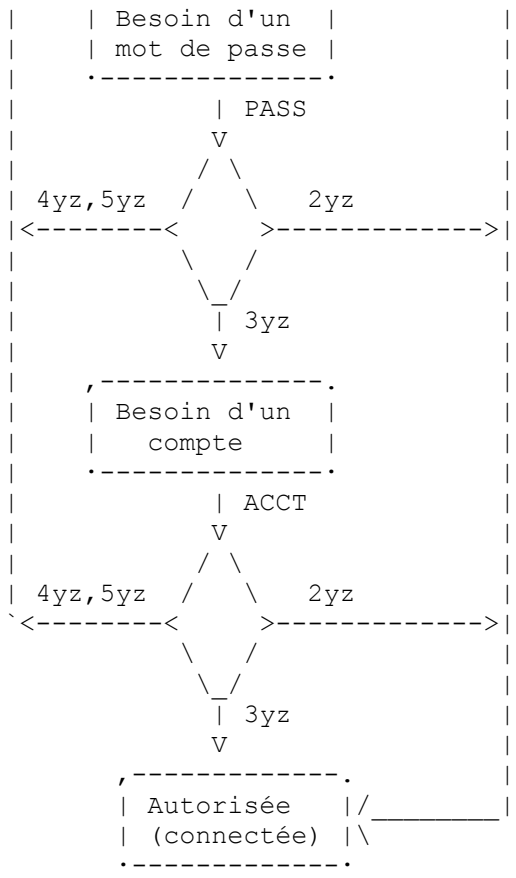
534, 535
RETR
534, 535
LIST
534, 535
NLST
534, 535
APPE
534, 535

En plus de ces codes de réponse, toute commande de sécurité peut retourner 500, 501, 502, 533, ou 421. Toute commande ftp peut retourner un code de réponse encapsulé dans une réponse 631, 632, ou 633 une fois qu'un échange de données de sécurité a été terminé avec succès.

9. Diagrammes d'états

Cette section comporte un diagramme d'états qui montre le flux d'authentification et d'autorisation dans une mise en œuvre de FTP à sécurité améliorée. Les blocs rectangulaires montrent les états où le client doit produire une commande, et les blocs en losanges montrent les états où le serveur doit produire une réponse.





10. Codage en base 64

Le codage base 64 est le même que le codage imprimable décrit au paragraphe 4.3.2.4 de la [RFC1421], excepté que les coupures de ligne ne doivent pas être incluses. Ce codage est défini comme suit.

En procédant de gauche à droite, la chaîne binaire résultant du programme de protection spécifique du mécanisme est codée en caractères qui sont universellement représentables sur tous les sites, bien que pas nécessairement avec les mêmes schémas binaires (par exemple, bien que le caractère "E" soit représenté dans un système fondé sur l'ASCII par l'hexadécimal 45 et par l'hexadécimal C5 dans un système fondé sur EBCDIC, la signification locale des deux représentations est équivalente).

On utilise un sous-ensemble à 64 caractères de l'alphabet international IA5, ce qui permet de représenter 6 bits par caractère imprimable. (Le sous-ensemble de caractères proposé est représenté de façon identique en IA5 et en ASCII.) Le caractère "=" signifie une fonction de traitement spéciale utilisée pour le bourrage au sein de la procédure de codage imprimable.

Le processus de codage représente des groupes de 24 bits d'entrée comme des chaînes en sortie de quatre caractères codés. Le traitement de gauche à droite sur un groupe d'entrée de 24 bits donne en sortie, à partir de la procédure de protection de message spécifique du mécanisme de sécurité, que chaque groupe de 6 bits est utilisé comme un indice dans une matrice de 64 caractères imprimables, à savoir "[A-Z][a- z][0-9]+/". Le caractère référencé par l'indice est placé dans la chaîne de sortie. Ces caractères sont choisis de façon à être universellement représentables, et l'ensemble exclut les caractères avec une signification particulière pour Telnet (par exemple, "<CR>", "<LF>", IAC).

Un traitement spécial est effectué si moins de 24 bits sont disponibles dans un groupe d'entrée à la fin d'un message. Un quantum de codage entier est toujours complété à la fin d'un message. Lorsque moins de 24 bits d'entrée sont disponibles dans un groupe d'entrée, des bits zéro sont ajoutés (sur la droite) pour former un nombre entier de groupes de 6 bits. Les positions de caractères de sortie qui ne sont pas nécessaires pour représenter des données d'entrée réelles sont réglées au caractère "=". Comme toute sortie codée canoniquement est un nombre entier d'octets, seuls les cas suivants peuvent survenir : (1) le quantum final d'entrées de codage est un multiple entier de 24 bits ; ici, l'unité finale de sortie codée sera un multiple entier de quatre caractères sans bourrage de "=", (2) le quantum final d'entrée codée est exactement de 8 bits ; ici, l'unité finale de sortie codée sera deux caractères suivis par deux caractères "=" de bourrage, ou (3) le quantum final d'entrée codée est exactement de 16 bits ; ici, l'unité finale de sortie codée sera trois caractères suivis par un caractère "=" de bourrage.

Les mises en œuvre doivent garder à l'esprit que le codage base 64 dans les commandes ADAT, MIC, CONF, et ENC, et dans les réponses 63z peut être de longueur arbitraire. Donc, la ligne entière doit être lue avant qu'elle puisse être traitée. Plusieurs lectures successives sur le canal de contrôle peuvent être nécessaires. Il n'est pas approprié qu'un serveur rejette une commande qui contient un codage base 64 simplement parce qu'il est trop long (en supposant que le décodage est par ailleurs bien formé dans le contexte dans lequel il a été envoyé).

La casse ne doit pas être ignorée lors de la lecture des commandes et réponses qui contiennent des codages en base 64.

11. Considérations pour la sécurité

Ce document est tout entier consacré aux considérations pour la sécurité qui se rapportent au protocole de transfert de fichier.

Les transferts de fichiers de tiers ne peuvent pas être sécurisés en utilisant ces extensions, car un contexte de sécurité ne peut pas être établi entre deux serveurs qui utilisent ces facilités (il n'existe pas de connexion de contrôle entre deux serveurs sur laquelle passer des jetons ADAT). Les travaux complémentaires dans ce domaine sont remis à une date ultérieure.

12. Remerciements

Je tiens à remercier les membres du groupe de travail CAT, ainsi que tous les participants aux discussions sur la liste de diffusion "cat-ietf@mit.edu", pour leurs contributions à ce document. Je tiens à remercier tout particulièrement Sam Sjogren, John Linn, Ted Ts'o, Jordan Brown, Michael Kogut, Derrick Brashear, John Gardiner Myers, Denis Pinkas, et Karri Balk pour leurs contributions à ce travail. Bien sûr, sans Steve Lunt, auteur de la première des six révisions de ce document, je n'existerai même pas.

13. Références

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

[RFC1123] R. Braden, éditeur, "Exigences pour les hôtes Internet – Application et prise en charge", STD 3, octobre 1989.

[RFC1421] J. Linn, "Amélioration de la confidentialité pour la messagerie électronique Internet : Partie I : Chiffrement de message et procédures d'authentification", février 1993. *(Historique)*

[RFC2941] T. Ts'o, éd., J. Altman, "Option d'authentification Telnet", septembre 2000. *(P.S.)*

14. Adresse de l'auteur

Marc Horowitz
Cygnus Solutions
955 Massachusetts Avenue
Cambridge, MA 02139
USA
téléphone : +1 617 354 7688
mél : marc@cygnus.com

Appendice I Spécification sous GSSAPI

Afin de maximiser l'utilité des nouveaux mécanismes de sécurité, il est souhaitable que ces nouveaux mécanismes soient mis en œuvre comme mécanismes GSSAPI plutôt que comme mécanismes de sécurité FTP. Cela permettra aux mises en œuvre existantes de FTP de prendre en charge plus facilement les nouveaux mécanismes, car il ne sera nécessaire de changer que peu de code, ou pas du tout. De plus, le mécanisme sera utilisable par d'autres protocoles, tels que IMAP, qui sont construits par dessus la GSSAPI, sans travail de spécification ou mise en œuvre supplémentaire nécessaire pour les concepteurs du mécanisme.

Le nom du mécanisme de sécurité (pour la commande AUTH) associé à tous les mécanismes qui emploient la GSSAPI est GSSAPI. Si le serveur prend en charge un mécanisme de sécurité qui emploie la GSSAPI, il doit répondre avec un code 334 qui indique qu'on attend ensuite une commande ADAT.

Le client doit commencer l'échange d'authentification en invoquant GSS_Init_Sec_Context, en passant 0 en input_context_handle (initial) et un targ_name *(nom de cible)* égal à output_name *(nom de sortie)* à partir du GSS_Import_Name *(nom d'entrée GSS)* invoqué avec input_name_type *(type de nom d'entrée)* du service fondé sur l'hôte *(Host-Based Service)* et input_name_string *(chaîne de nom d'entrée)* de "ftp@hostname" où "hostname" est le nom d'hôte pleinement qualifié du serveur avec toutes les lettres en minuscules. (À défaut de quoi le client peut réessayer en utilisant la input_name_string de "host@hostname".) Le jeton de sortie *(output_token)* doit alors être codé en base 64 et envoyé au serveur comme l'argument d'une commande ADAT. Si GSS_Init_Sec_Context retourne GSS_S_CONTINUE_NEEDED, le client doit alors s'attendre à ce qu'un jeton soit retourné dans la réponse à la commande ADAT. Ce jeton doit ensuite être passé à une autre invocation à GSS_Init_Sec_Context. Dans ce cas, si GSS_Init_Sec_Context ne retourne pas un jeton de sortie, le code de réponse provenant du serveur pour la précédente commande ADAT doit avoir été 235. Si GSS_Init_Sec_Context retourne GSS_S_COMPLETE, aucun autre jeton n'est attendu de la part du serveur, et le client doit considérer le serveur comme authentifié.

Le serveur doit décoder en base 64 l'argument de la commande ADAT et passer le jeton résultant au GSS_Accept_Sec_Context comme jeton d'entrée, en réglant "acceptor_cred_handle" à NULL (pour "utiliser les accréditifs par défaut") et 0 pour input_context_handle (initial). Si un jeton de sortie *(output_token)* est retourné, il doit être codé en base 64 et retourné au client en incluant "ADAT=base64string" dans le texte de la réponse. Si GSS_Accept_Sec_Context retourne GSS_S_COMPLETE, le code de réponse doit être 235, et le serveur doit considérer que le client est authentifié. Si GSS_Accept_Sec_Context retourne GSS_S_CONTINUE_NEEDED, le code de réponse doit être 335. Autrement, le code de réponse devrait être 535, et le texte de la réponse devrait contenir un message d'erreur descriptif.

L'entrée chan_bindings à GSS_Init_Sec_Context et à GSS_Accept_Sec_Context devrait utiliser l'adresse Internet du client et l'adresse Internet du serveur comme, respectivement, l'adresse de l'initiateur et de l'acceptant. Le type d'adresse devrait être pour les deux GSS_C_AF_INET. Aucune données d'application ne devrait être spécifiée.

Comme GSSAPI accepte des homologues anonymes pour des contextes de sécurité, il est possible que l'authentification du client auprès du serveur n'établisse pas en fait une identité.

La procédure associée aux commandes MIC, aux réponses 631, et aux transferts sûrs de fichiers est :

GSS_Wrap pour l'envoyeur, avec conf_flag == FAUX
GSS_Unwrap pour le receveur

La procédure associée à la commande ENC, aux réponses 632, et aux transferts de fichiers privés est :

GSS_Wrap pour l'envoyeur, avec conf_flag == VRAI
GSS_Unwrap pour le receveur

Les commandes CONF et les réponses 633 ne sont pas acceptées.

Le client et le serveur devraient tous deux inspecter la valeur de conf_avail pour déterminer si l'homologue prend en charge les services de confidentialité.

Lorsque l'état de sécurité est restitué (lorsque AUTH est reçu pour la seconde fois, ou lorsque REIN est reçu) cela devrait être fait en invoquant la fonction GSS_Delete_sec_context.

Appendice II Spécification sous Kerberos version 4

Le nom du mécanisme de sécurité (pour la commande AUTH) associée avec Kerberos version 4 est KERBEROS_V4. Si le serveur prend en charge KERBEROS_V4, il doit répondre avec le code 334 qui indique qu'une commande ADAT est attendue ensuite.

Le client doit restituer un ticket pour le principal Kerberos "ftp.nom_d'hôte@domaine" en invoquant krb_mk_req(3) avec un nom de principal de "ftp", une instance égale à la première partie du nom d'hôte canonique du serveur avec toutes les lettres en minuscules (comme retourné par krb_get_phost(3)) le nom de domaine du serveur (comme retourné par krb_realmofhost(3)) et une somme de contrôle arbitraire. Le ticket doit alors être codé en base 64 et envoyé comme

argument d'une commande ADAT.

Si le nom principal "ftp" n'est pas un principal enregistré dans la base de données Kerberos, le client peut alors retomber sur le nom principal "rcmd" (même instance et domaine). Cependant, les serveurs doivent n'accepter que l'un ou l'autre de ces noms principaux, et ne doivent pas vouloir accepter l'un et l'autre. Généralement, si le serveur a une clé pour le principal "ftp" dans son srvtab, ce principal seul doit alors être utilisé, autrement, seul le principal "rcmd" doit être utilisé.

Le serveur doit décoder en base 64 l'argument de la commande ADAT et passer le résultat à krb_rd_req(3). Le serveur doit ajouter un à la somme de contrôle provenant de l'authentificateur, convertir le résultat à l'ordre des octets du réseau (octet de poids fort en premier) et le signer en utilisant krb_mk_safe(3) et coder le résultat en base 64. En cas de succès, le serveur doit répondre au client avec un code 235 et inclure "ADAT=base64string" dans le texte de la réponse. En cas d'échec, le serveur devrait réponde 535.

À réception de la réponse 235 du serveur, le client doit analyser le texte de la réponse à la recherche des données codées en base 64, les décoder, les convertir de l'ordre des octets du réseau, et passer le résultat à krb_rd_safe(3). Le client doit considérer que le serveur est authentifié si la somme de contrôle résultante est égale à un plus la valeur envoyée précédemment.

La procédure associée aux commandes MIC, aux réponses 631, et aus transferts de fichiers sûrs est :

krb_mk_safe(3) pour l'envoyeur
krb_rd_safe(3) pour le receveur

La procédure associée aux commandes ENC, aux réponses 632, et aux tranferts de fichiers privés est :

krb_mk_priv(3) pour l'envoyeur
krb_rd_priv(3) pour le receveur

Les commandes CONF et les réponses 633 ne sont pas acceptées.

Noter que pour KERBEROS_V4, la présente spécification ne contient aucune disposition pour négocier des moyens de remplacement pour les progerammes d'intégrité et de confidentialité. Noter aussi que l'échange ADAT ne couvre paas la question de savoir si l'homologue prend en charge les services de confidentialité.

Pour rester dans la PBSZ admise, les développeurs doivent noter qu'une mémoire tampon de texte en clair va augmenter de 31 octets lors du traitement par krb_mk_safe(3) et de 26 octets lors du traitement par krb_mk_priv(3).

Déclaration complète de droits de reproduction