Groupe de travail Réseau
**Request for Comments : 1961**
Catégorie : En cours de normalisations

P. McMahon, ICL
juin 1996
Traduction Claude Brière de L'Isle

# Méthode d'authentification de GSS-API pour SOCKS version 5

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

## Table des Matières



## 1. Objet

La spécification du protocole pour SOCKS version 5 [RFC1928] donne un cadre généralisé pour l'utilisation de protocoles d'authentification arbitraires dans l'établissement initial de la connexion SOCKS. Le présent document fournit la spécification pour le protocole d'authentification GSS-API de SOCKS v5, et définit une encapsulation fondée sur GSS-API pour la fourniture de l'authentification, la protection de l'intégrité, et facultativement, de la confidentialité.

## 2. Introduction

GSS-API fournit une interface abstraite qui assure des services de sécurité à utiliser dans des applications réparties, mais isole les appelants des mécanismes et mises en œuvre de sécurité spécifiques.

Les homologues GSS-API réalisent l'interopérabilité par l'établissement d'un mécanisme de sécurité commun pour l'établissement d'un contexte de sécurité – soit par une action administrative, soit par une négociation. GSS-API est spécifié dans la [RFC1508] et la [RFC1509]. La présente spécification est destinée à être utilisée avec des mises en œuvre de GSS-API, et de la spécification émergeante GSS-API v2.

L'approche de l'utilisation de GSS-API dans SOCKS v5 est d'authentifier le client et le serveur en réussissant à établir un

contexte de sécurité GSS-API de telle sorte que GSS-API encapsule tout protocole de négociation pour le choix du mécanisme, et l'accord sur les options du service de sécurité.

La GSS-API permet à l'initiateur du contexte de savoir quels services de sécurité la cible prend en charge pour le mécanisme choisi. Le niveau de protection requis est alors défini par négociation.

Des invocations de protection GSS-API message par message sont ultérieurement utilisées pour encapsuler tout trafic TCP et UDP ultérieur entre client et serveur.

## 3. Établissement de contexte de sécurité GSS-API

### 3.1 Préparation

Avant l'utilisation des primitives GSS-API, client et serveur devraient être authentifiés localement, et avoir établi des accréditifs GSS-API par défaut.

Le client devrait invoquer gss_import_name pour obtenir une représentation interne du nom du serveur. Pour une portabilité maximale, le type de nom par défaut GSS_C_NULL_OID devrait être utilisé pour spécifier l'espace de noms par défaut, et l'entrée name_string devrait être traitée par le code du client comme une entrée opaque spécifique de l'espace de noms.

Par exemple, lorsque on utilise la désignation Kerberos v5, le nom importé peut être de la forme "SERVICE:socks@nom_d'hôte_de_serveur_socks" où "nom_d'hôte_de_serveur_socks" est le nom d'hôte pleinement qualifié du serveur avec toutes ses lettres en minuscules. D'autres mécanismes peuvent, cependant, avoir une forme de nom différente, de sorte que le client ne devrait pas faire d'hypothèses sur la syntaxe du nom.

### 3.2 Établissement de contexte de client

Le client devrait alors invoquer gss_init_sec_context, passant normalement :

GSS_C_NO_CREDENTIAL en cred_handle pour spécifier l'accréditif par défaut (pour l'usage de l'initiateur)

GSS_C_NULL_OID en mech_type pour spécifier le mécanisme par défaut,

GSS_C_NO_CONTEXT en context_handle pour spécifier un contexte (initialement) NUL, et,

le nom de serveur importé précédemment dans nom_de_cible.

Le client DOIT aussi spécifier ses exigences en matière de protection contre la répétition, de délégation, et de protection de séquence via le paramètre req_flags gss_init_sec_context. La présente spécification exige que le client demande toujours ces options de service (c'est-à-dire, passe GSS_C_MUTUAL_FLAG | GSS_C_REPLAY_FLAG | GSS_C_DELEG_FLAG | GSS_C_SEQUENCE_FLAG dans req_flags).

Cependant, GSS_C_SEQUENCE_FLAG ne devrait être passé que pour les clients fondés sur TCP, et pour les clients fondés sur UDP.

### 3.3 Codes d'état majeurs d'établissement de contexte de client

Le code d'état gss_init_sec_context retourné peut prendre deux valeurs de succès différentes :

- Si gss_init_sec_context retourne GSS_S_CONTINUE_NEEDED, le client devrait alors s'attendre à ce que le serveur produise un jeton dans la réponse de sous négociation qui va suivre. Le client DOIT passer le jeton à une autre invocation de gss_init_sec_context, et répéter cette procédure jusqu'à ce que les opérations "continue" soient achevées.

- Si gss_init_sec_context retourne GSS_S_COMPLETE, le client devrait alors répondre au serveur avec tout jeton de résultat obtenu.

Si il n'y a pas de jeton de résultat, le client devrait envoyer les détails de la demande protégée, incluant toute sous négociation de protection de message requise comme spécifié aux Sections 4 et 5.

### 3.4 Jeton initial de client

La mise en œuvre GSS-API du client répond alors normalement avec le jeton de résultat que le client envoie au serveur dans un message.

```
+-------+------+--------+.....................  .+
+version| mtyp |longueur|       jeton             |
+-------+------+--------+..................  .....+
+ 0x01  | 0x01 |  0x02  | jusqu'à 2^16 - 1 octets |
+-------+------+--------+.....................  ..+
```

Où :
- "version" est le numéro de version du protocole ; ici 1 pour représenter la première version de SOCKS/GSS-API,
- "mtyp" est le type de message , ici 1 pour représenter un message d'authentification,
- "longueur" est la longueur du champ "jeton" en octets,
- "jeton" est le jeton d'authentification opaque émis par GSS-API.

### 3.5 Échec d'initialisation de client GSS-API

Si, cependant, la mise en œuvre GSS-API du client échoue durant gss_init_sec_context, le client DOIT clore sa connexion avec le serveur.

### 3.6 Établissement de contexte de serveur

Dans le cas où un client a envoyé avec succès un jeton émis par gss_init_sec_context() au serveur, le serveur DOIT passer à gss_accept_sec_context le jeton fourni par le client comme jeton_d'entrée.

Lorsque il invoque gss_accept_sec_context() pour la première fois, l'argument context_handle est initialement réglé à GSS_C_NO_CONTEXT.

Pour la portabilité, verifier_cred_handle est réglée à GSS_C_NO_CREDENTIAL pour spécifier les accréditifs par défaut (pour l'usage de l'accepteur).
- Si gss_accept_sec_context retourne GSS_CONTINUE_NEEDED, le serveur devrait retourner au client le jeton_de_résultat généré, et passer ensuite le jeton résultant fourni par le client dans une autre invocation à gss_accept_sec_context.
- Si gss_accept_sec_context retourne GSS_S_COMPLETE, alors, si un jeton_de_résultat est retourné, le serveur devrait le retourner au client.
- Si aucun jeton n'est retourné, un jeton de longueur zéro devrait être envoyé par le serveur pour signaler au client qu'il est prêt à recevoir la demande du client.

### 3.7 Réponse du serveur

Dans tous les cas de continuation/confirmation, le serveur utilise le même type de message que pour l'interaction client -> serveur.

```
+-------+------+--------+.....................  .+
+version| mtyp |longueur|       jeton             |
+-------+------+--------+..................  .....+
+ 0x01  | 0x01 |  0x02  | jusqu'à 2^16 - 1 octets |
+-------+------+--------+.....................  ..+
```

### 3.8 Échec de contexte de sécurité

Si le serveur refuse la connexion du client pour n'importe quelle raison (échec d'authentification GSS-API ou autre), il va retourner :

```
+-------+------+
+version| mtyp |
+-------+------+
+  0x01 | 0xff |
+-------+------+
```

Où :
- "version" est le numéro de version du protocole ; ici 1 pour représenter la première version de SOCKS/GSS-API,
- "mtyp" est le type de message, ici 0xff pour représenter un message d'interruption.

## 4. Options de niveau de protection GSS-API

### 4.1 Protection de message

L'établissement d'un contexte de sécurité GSS-API permet aux homologues communicants de déterminer quels services de protection par message leurs sont disponibles au moyen des fanions ret_flags GSS_C_INTEG_FLAG et GSS_C_CONF_FLAG de gss_init_sec_context() et gss_accept_sec_context() qui indiquent respectivement les services d'intégrité et de confidentialité de message.

Il est nécessaire de s'assurer que la protection de message appliquée au trafic est appropriée à la sensibilité des données, et à la sévérité des menaces.

### 4.2 Sous négociation de protection de message

Pour les clients et serveurs TCP et UDP, différent niveaux de protection sont possibles dans le protocole SOCKS v5, de sorte qu'une étape de sous négociation supplémentaire est nécessaire pour se mettre d'accord sur le niveau de protection du message. Après l'achèvement réussi de cette sous négociation, les clients et serveurs TCP et UDP utilisent l'encapsulation GSS-API comme défini au paragraphe 5.1.

Après la réussite de l'établissement d'un contexte de sécurité GSS-API, La mise en œuvre GSS-API du client envoie son niveau de protection de contexte de sécurité requis au serveur. Le serveur retourne alors le niveau de protection de contexte de sécurité avec lequel il est en accord – qui peut tenir compte ou non de la demande du client.

Le niveau de protection de contexte de sécurité envoyé par le client et le serveur DOIT avoir une des valeurs suivantes :
1 intégrité par message exigée
2 intégrité et confidentialité par message exigée
3 intégrité ou confidentialité sélective par message sur la base de la configuration locale du client et du serveur.

On pense que la plupart des mises en œuvre vont se mettre d'accord sur le niveau 1 ou 2 à cause de la difficulté pratique d'appliquer des contrôles sélectifs aux messages qui passent au travers d'une bibliothèque socks.

### 4.3 Format de message de sous négociation de protection de message

Le niveau de protection de contexte de sécurité est envoyé du client au serveur et vice versa en utilisant le format de message protégé suivant :

```
+-------+------+--------+.....................  .+
+version| mtyp |longueur|       jeton             |
+-------+------+--------+..................  .....+
+ 0x01  | 0x02 |  0x02  | jusqu'à 2^16 - 1 octets |
+-------+------+--------+.....................  ..+
```

Où :
- "version" est le numéro de version du protocole ; ici 1 pour représenter la première version de SOCKS/GSS-API,
- "mtyp" est le type de message, ici 2 pour représenter un message de négociation de niveau de protection,
- "longueur" est la longueur du champ "jeton" en octets.

### 4.4 Génération de message de sous négociation de protection de message

Le jeton est produit en encapsulant un octet contenant le niveau de protection requis en utilisant gss_seal()/gss_wrap() avec conf_req réglé à FAUX. Le jeton est vérifié en utilisant gss_unseal()/gss_unwrap().

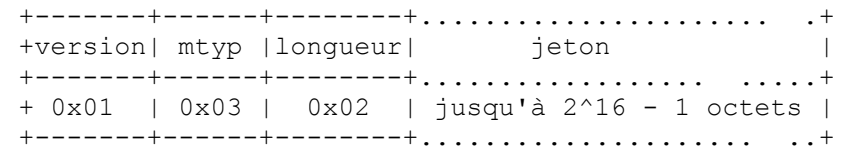
Si le choix du serveur du niveau de protection est inacceptable au client, celui-ci DOIT clore sa connexion au serveur.

## 5. Protection par message GSS-API

Pour les clients et serveurs TCP et UDP, les fonctions GSS-API d'encapsulation et de désencapsulation devront être utilisées par les mises en œuvre - c'est-à-dire, gss_seal()/gss_wrap(), et gss_unseal()/ gss_unwrap().

La valeur par défaut de la qualité de protection devra être spécifiée, et l'utilisation de conf_req_flag devra être comme déterminé par l'étape de sous négociation précédente. Si le niveau de protection 1 est retenu, conf_req_flag DOIT alors toujours être FAUX ; si le niveau de protection 2 est retenu, alors conf_req_flag DOIT toujours être VRAI ; et si le niveau de protection 3 est retenu, alors conf_req est déterminé message par message par le client et le serveur en utilisant la configuration locale.

Tous les messages encapsulés sont munis en préfixe de la trame suivante :




Où :
- "version" est le numéro de version du protocole ; ici 1 pour représenter la première version de SOCKS/GSS-API,
- "mtyp" est le type de message, ici 3 pour représenter les données encapsulées d'utilisateur,
- "longueur" est la longueur du champ "jeton" en octets,
- "jeton" sont les données d'utilisateur encapsulées par GSS-API.

## 6. Terminaison du contexte de sécurité GSS-API

Le message de terminaison de contexte GSS-API (émis par gss_delete_sec_context) n'est pas utilisé par ce protocole.

Lorsque la connexion est close, chaque homologue invoque gss_delete_sec_context() passant GSS_C_NO_BUFFER dans l'argument jeton_de_résultat.

## 7. Références

[RFC1508] J. Linn, "Interface générique de programme de service de sécurité", septembre 1993. *(P.S.) (remplacée par* RFC2743)

[RFC1509] J. Wray, "API de service générique de sécurité : liens C", septembre 1993. *(remplacée par* RFC2744)

[RFC1928] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas et L. Jones, "Protocole SOCKS version 5", mars 1996.

## 8. Remerciements

Le présent document s'appuie sur un mémoire précédent produit par Marcus Leech (BNR) – dont les commentaires ont été les bienvenus. Il reflète aussi les apports du groupe de travail AFT, et les commentaires découlant de l'expérience de mise en œuvre de Xavier Gosselin (IUT Lyons).

## 9. Considérations pour la sécurité

Les services de sécurité fournis par GSS-API dépendent entièrement de l'efficacité des mécanismes de sécurité sous-jacents et de la correction de la mise en œuvre des algorithmes et protocoles sous-jacents.

L'utilisateur d'un service GSS-API doit s'assurer que la qualité de protection fournie par la mise en œuvre du mécanisme est cohérente avec sa politique de sécurité.

De plus, lorsque la négociation est prise en charge sous GSS-API, des contraintes sur les mécanismes acceptables peuvent être imposées pour assurer qu'ils conviennent à l'application à la traversée des pare-feu authentifiés.

## 10. Adresse de l'auteur

P. V. McMahon
ICL Enterprises
Kings House
33 Kings Road
Reading, RG1 3PX
UK

mél : p.v.mcmahon@rea0803.wins.icl.co.uk
téléphone : +44 1734 634882
fax : +44 1734 855106