Groupe de travail Réseau
Request for Comments : 1928
Categorie : En cours de normalisation
mars 1996

Traduction Claude Brière de L'Isle

M. Leech, Bell-Northern Research Ltd
M. Ganis, International Business Machines
Y. Lee, NEC Systems Laboratory
R. Kuris, Unify Corporation
D. Koblas, Independent Consultant
L. Jones, Hewlett-Packard Company

Protocole 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.

Remerciements
Le présent mémoire décrit un protocole qui est une évolution de la précédente version du protocole, la version 4 [1]. Ce nouveau protocole découle de discussions actives et de la mise en œuvre de prototypes. Les contributeurs clés sont Marcus Leech de Bell-Northern Research, David Koblas, consultant indépendant, Ying-Da Lee de NEC Systems Laboratory, LaMont Jones de Hewlett-Packard Company, Ron Kuris de Unify Corporation, Matt Ganis de International Business Machines.

1. Introduction

L'utilisation de pare-feu de réseau, des systèmes qui isolent efficacement la structure du réseau interne d'une organisation d'un réseau extérieur, comme l'Internet, devient de plus en plus populaire. Ces systèmes de pare-feu agissent normalement comme des passerelles de couche application entre les réseaux, offrant usuellement un accès contrôlé à TELNET, FTP, et SMTP. Avec l'émergence de protocoles de couche d'application plus sophistiqués conçus pour faciliter la découverte d'informations mondiales, il existe un besoin de fournir un cadre général pour que ces protocoles traversent un pare-feu en toute transparence et sécurité.

Il existe aussi un besoin d'authentification forte d'une telle traversée d'une granularité aussi fine que possible. Cette exigence découle de la réalisation que la relation client-serveur émerge entre les réseaux de diverses organisations, et que de telles relations doivent être contrôlées et souvent fortement authentifiées.

Le protocole décrit ici est conçu pour fournir un cadre pour les applications client-serveur dans les deux domaines TCP et UDP pour utiliser de façon pratique et sûre les services d'un pare-feu réseau. Le protocole est conçu comme une "couche d'ajustement" entre la couche Application et la couche Transport, et à ce titre ne produit pas de services de passerelle de couche Réseau, comme la transmission des messages ICMP.

2. Pratiques existantes

Il existe actuellement un protocole, SOCKS version 4, qui fournit une traversée de pare-feu non sécurisée pour les applications client-serveur fondées sur TCP, incluant TELNET, FTP et les protocoles populaires de découverte d'information comme HTTP, WAIS et GOPHER.

Ce nouveau protocole étend le modèle de SOCKS version 4 pour y inclure UDP, il étend le cadre pour y inclure des dispositions pour généraliser des schémas d'authentification forte, et il étend le schéma d'adressage pour englober les noms de domaine et les adresses IPv6.

La mise en œuvre du protocole SOCKS implique normalement la recompilation ou le reformatage des applications client-serveur fondées sur TCP pour utiliser les sous programmes d'encapsulation appropriés dans la bibliothèque de SOCKS.

Note : sauf mention contraire, les nombres décimaux qui apparaissent dans les diagrammes de format de paquet représentent la longueur du champ correspondant, en octets. Lorsque un certain octet doit prendre une valeur spécifique, la syntaxe X'hh' est utilisée pour noter la valeur du seul octet de ce champ. Lorsque le mot 'Variable' est utilisé, il indique que le champ correspondant a une longueur variable définie par un champ de longueur associé (un ou deux octets) ou par un champ de type de données.

3. Procédure pour les clients fondés sur TCP

Lorsque un client fondé sur TCP souhaite établir une connexion avec un objet qui n'est accessible que via un pare-feu (une telle détermination est du ressort de la mise en œuvre) il doit ouvrir une connexion TCP à l'accès SOCKS approprié sur le système serveur SOCKS. Le service SOCKS est conventionnellement situé à l'accès TCP 1080. Si la demande de connexion réussit, le client entre en négociation pour la méthode d'authentification à utiliser, il s'authentifie avec la méthode choisie, puis envoie une demande de relais. Le serveur SOCKS évalue la demande, et soit établit la connexion appropriée, soit la refuse.

Note : sauf mention contraire, les nombres décimaux qui apparaissent dans les diagrammes de format de paquet représentent la longueur du champ correspondant, en octets. Lorsque un certain octet doit prendre une valeur spécifique, la syntaxe X'hh' est utilisée pour noter la valeur du seul octet de ce champ. Lorsque le mot 'Variable' est utilisé, il indique que le champ correspondant a une longueur variable définie par un champ de longueur associé (un ou deux octets) ou par un champ de type de données.

Le client se connecte au serveur, et envoie un message de choix de méthode/identifiant de version :

```
+----+-----------+----------+
|VER | NMETHODES | METHODES |
+----+-----------+----------+
| 1  |    1      |  1 à 255 |
+----+-----------+----------+
```

Le champ VER est réglé à X'05' pour la présente version du protocole. Le champ NMETHODES contient le nombre d'octets d'identifiant de méthode qui apparaissent dans le champ METHODES.

Le serveur choisit l'une des méthodes données dans METHODES, et envoie un message de choix de METHODE :

```
+----+---------+
|VER | METHODE |
+----+---------+
| 1  |   1     |
+----+---------+
```

Si la METHODE choisie est X'FF', aucune des méthodes citées par le client n'est acceptable, et le client DOIT clore la connexion.

Les valeurs actuellement définies pour METHODE sont :
- X'00' : pas d'authentification exigée
- X'01' : GSSAPI
- X'02' : nom d'utilisateur/mot de passe
- X'03' à X'7F' : alloué par l'IANA
- X'80' à X'FE' : réservé pour des méthodes privées
- X'FF' : pas de méthode acceptable

Le client et le serveur entrent alors dans une sous négociation spécifique de la méthode.
Les descriptions des sous négociations selon la méthode apparaissent dans des documents séparés.

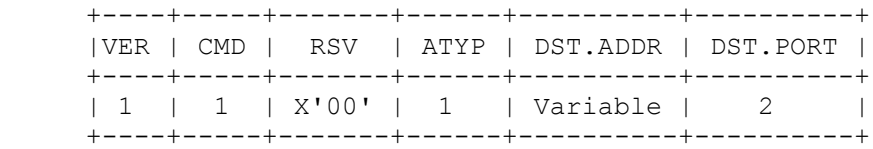
Les développeurs de nouveaux supports de méthodes pour le présent protocole devraient contacter l'IANA pour obtenir un numéro de METHODE. Le document d'allocation des numéros de l'Internet devrait être mentionné comme détenant la liste actuelle des numéros de méthode et de leurs protocoles correspondants.

Les mises en œuvre conformes DOIVENT prendre en charge GSSAPI et DEVRAIENT prendre en charge les méthodes d'authentification par nom d'utilisateur/mot de passe.

4. Demandes

Une fois achevée la sous négociation dépendante de la méthode, le client envoie les détails de la demande. Si la méthode négociée inclut une encapsulation pour les besoins de vérification d'intégrité et/ou de confidentialité, ces demandes DOIVENT être encapsulées dans l'encapsulation définie par la méthode.

La demande SOCKS est formée comme suit :



VER	CMD	RSV	ATYP	DST.ADDR	DST.PORT
1	1	X'00'	1	Variable	2

où :
- VER : version du protocole : X'05'
- CMD : commande
  - X'01' CONNECT
  - X'02' BIND
  - X'03' UDP ASSOCIATE
- RSV : réservé
- ATYP : type d'adresse de l'adresse qui suit :
  - X'01' adresse IPv4
  - X'03' nom de domaine
  - X'04' adresse IPv6
- DST.ADDR : adresse de destination désirée
- DST.PORT : accès de destination désiré dans l'ordre d'octet du réseau.

Le serveur SOCKS va normalement évaluer la demande sur la base des adresses de source et de destination, et retourner un ou plusieurs messages de réponse, comme approprié pour le type de demande.

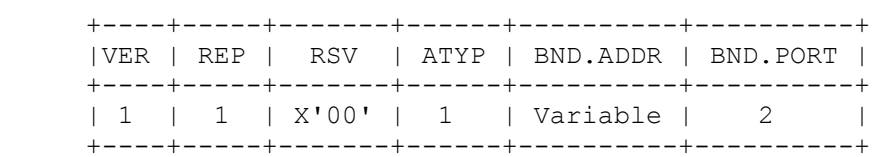
5. Adressage

Dans un champ d'adresse (DST.ADDR, BND.ADDR),le champ ATYP spécifie le type d'adresse contenu dans le champ :

- X'01' : l'adresse est IP version 4, d'une longueur de 4 octets
- X'03' : l'adresse contient un nom de domaine pleinement qualifié. Le premier octet du champ d'adresse contient le nombre d'octets du nom qui suit ; il n'y a pas d'octet NUL en terminaison.
- X'04' : l'adresse est IP version 6, d'une longueur de 16 octets.

6. Réponses

Les informations de demande SOCKS sont envoyées par le client aussitôt qu'il a établi une connexion avec le serveur SOCKS, et achevé les négociations d'authentification. Le serveur évalue la demande, et retourne une réponse formée comme suit :



VER	REP	RSV	ATYP	BND.ADDR	BND.PORT
1	1	X'00'	1	Variable	2

où :
- VER : X'05' : version du protocole
- REP : champ de réponse
  - X'00' succès
  - X'01' échec général du serveur SOCKS
  - X'02' connexion interdite par les règles
  - X'03' réseau injoignable
  - X'04' hôte injoignable
  - X'05' connexion refusée
  - X'06' TTL expiré
  - X'07' commande non acceptée
  - X'08' type d'adresse non accepté
  - X'09' à X'FF' non alloué
- RSV ; réservé
- ATYP : type d'adresse de l'adresse qui suit
  - X'01' adresse IPv4

X'03' nom de domaine
X'04' adresse IPv6

- BND.ADDR : adresse du serveur lié
- BND.PORT : accès du serveur lié dans l'ordre d'octet du réseau

Les champs marqués réservé (RSV) doivent être réglés à X'00'.

Si la méthode choisie inclut l'encapsulation pour des besoins d'authentification, d'intégrité et/ou de confidentialité, les réponses sont encapsulées dans l'encapsulation dépendante de la méthode.

CONNECT
Dans la réponse à CONNECT, BND.PORT contient le numéro d'accès que le serveur a alloué pour connecter la cible à l'hôte, tandis que BND.ADDR contient l'adresse IP associée. La BND.ADDR fournie est souvent différente de l'adresse IP que le client utilise pour atteindre le serveur SOCKS, car de tels serveurs sont souvent multi rattachements. On s'attend à ce que le serveur SOCKS utilise DST.ADDR et DST.PORT, et l'adresse et l'accès de source côté client pour évaluer la demande CONNECT.

BIND
La demande BIND est utilisée dans les protocoles qui exigent que le client accepte les connexions provenant du serveur. FTP est un exemple bien connu, qui utilise la principale connexion client-serveur pour les commandes et les rapports d'état, mais peut utiliser une connexion serveur-client pour transférer des données à la demande (par exemple, LS, GET, PUT).

On s'attend à ce que le côté client d'un protocole d'application utilise la demande BIND seulement pour établir des connexions secondaires après l'établissement d'une connexion principale utilisant CONNECT. Il est prévu qu'un serveur SOCKS utilise DST.ADDR et DST.PORT pour évaluer la demande BIND.

Deux réponses sont envoyées du serveur SOCKS au client durant une opération BIND. La première est envoyée après que le serveur a créé et s'est lié à une nouvelle prise. Le champ BND.PORT contient le numéro d'accès que le serveur SOCKS alloue pour écouter les connexions entrantes. Le champ BND.ADDR contient l'adresse IP associée. Le client va normalement utiliser ces éléments d'informations pour notifier (via la connexion principale ou de contrôle) au serveur d'application l'adresse de rendez-vous. La seconde réponse ne survient qu'après que la connexion entrante anticipée a réussi ou échoué.

Dans la seconde réponse, les champs BND.PORT et BND.ADDR contiennent l'adresse et le numéro d'accès de l'hôte connecté.

UDP ASSOCIATE
La demande UDP ASSOCIATE est utilisée pour établir une association au sein du processus de relais UDP pour traiter les datagrammes UDP. Les champs DST.ADDR et DST.PORT contiennent l'adresse et l'accès que le client s'attend à utiliser pour envoyer les datagrammes UDP pour l'association. Le serveur PEUT utiliser ces informations pour limiter l'accès à l'association. Si le client n'est pas en possession des informations au moment de l'UDP ASSOCIATE, le client DOIT utiliser un numéro d'accès et une adresse tout à zéro.
Une association UDP se termine lorsque la connexion TCP sur laquelle est arrivée la demande UDP ASSOCIATE se termine.
Dans la réponse à une demande UDP ASSOCIATE, les champs BND.PORT et BND.ADDR indiquent le numéro d'accès et l'adresse à laquelle le client DOIT envoyer les messages de demande UDP à relayer.

Traitement de la réponse
Lorsque une réponse (valeur de REP autre que X'00') indique un échec, le serveur SOCKS DOIT terminer la connexion TCP peu de temps après avoir envoyé la réponse. Ce ne doit pas être plus de 10 secondes après la détection de la condition qui a causé l'échec.

Si le code de réponse (valeur de REP de X'00') indique un succès, et si la demande était un BIND ou un CONNECT, le client peut maintenant commencer à passer les données. Si la méthode d'authentification choisie accepte l'encapsulation pour les besoins de l'intégrité, de l'authentification et/ou la confidentialité, les données sont encapsulées en utilisant l'encapsulation dépendant de la méthode. De même, lorsque les données arrivent au serveur SOCKS pour le client, le serveur DOIT encapsuler les données comme approprié pour la méthode d'authentification utilisée.

7. Procédure pour les clients fondés sur UDP

Un client fondé sur UDP DOIT envoyer ses datagrammes au serveur relais UDP à l'accès UDP indiqué par BND.PORT dans la réponse à la demande UDP ASSOCIATE. Si la méthode d'authentification choisie fournit l'encapsulation pour les besoins de l'authenticité, l'intégrité, et/ou la confidentialité, le datagramme DOIT être encapsulé en utilisant l'encapsulation appropriée.

Chaque datagramme UDP porte avec lui un en-tête de demande UDP :

```
+----+------+------+----------+----------+----------+
|RSV | FRAG | ATYP | DST.ADDR | DST.PORT |   DATA   |
+----+------+------+----------+----------+----------+
| 2  |  1   |  1   | Variable |    2     | Variable |
+----+------+------+----------+----------+----------+
```

Les champs de l'en-tête de demande UDP sont :
- RSV : réservé X'0000'
- FRAG : numéro de fragment actuel
- ATYP : type d'adresse des adresses suivantes :
  - X'01' : adresse IPv4
  - X'03' : nom de domaine
  - X'04' : adresse IPv6
- DST.ADDR : adresse de destination désirée
- DST.PORT : accès de destination désiré
- DATA : données d'utilisateur

Lorsque un serveur relais UDP décide de relayer un datagramme UDP, il le fait en silence, sans aucune notification au client demandeur. De même, il va éliminer les datagrammes qu'il ne peut ou veut relayer. Lorsque un serveur relais UDP reçoit un datagramme de réponse d'un hôte distant, il DOIT l'encapsuler en utilisant l'en-tête de demande UDP ci-dessus, et toute encapsulation d'authentification dépendante de la méthode.

Le serveur relais UDP DOIT acquérir du serveur SOCKS l'adresse IP attendue du client qui va envoyer les datagrammes au BND.PORT donné dans la réponse à UDP ASSOCIATE. Il DOIT abandonner tout datagramme arrivant de toute adresse IP de source autre que celle enregistrée pour cette association particulière.

Le champ FRAG indique si ce datagramme est ou non inclus dans le nombre des fragments. Si il est mis en œuvre, le bit de poids fort indique la séquence de la fin du fragment, tandis qu'une valeur de X'00' indique que ce datagramme est solitaire. Les valeurs entre 1 et 127 indiquent la position du fragment au sein d'une séquence de fragments. Chaque receveur aura un REASSEMBLY QUEUE *(file d'attente de réassemblage)* et un REASSEMBLY TIMER *(temporisateur de réassemblage)* associés à ces fragments. La file d'attente de réassemblage doit être réinitialisée et les fragments associés abandonnés chaque fois que le temporisateur de réassemblage arrive à expiration, ou qu'un nouveau datagramme arrive portant un champ FRAG dont la valeur est inférieure à la plus forte valeur de FRAG traitée pour cette séquence de fragments. Le temporisateur de réassemblage NE DOIT PAS faire moins de 5 secondes. Il est recommandé que la fragmentation soit évitée chaque fois que possible par les applications.

La mise en œuvre de la fragmentation est facultative ; une mise en œuvre qui ne prend pas la fragmentation en charge DOIT éliminer tout datagramme dont le champ FRAG est autre que X'00'.

L'interface de programmation pour un UDP capable de SOCKS DOIT rapporter un espace de mémoire tampon disponible pour les datagrammes UDP qui soit inférieur à l'espace réel fourni par le système d'exploitation :
- si ATYP est X'01' – plus petit de 10 octets + selon la méthode
- si ATYP est X'03' – plus petit de 262 octets + selon la méthode
- si ATYP est X'04' – plus petit de 22 octets + selon la méthode

8. Considérations pour la sécurité

Le présent document décrit un protocole pour la traversée de la couche application des pare-feu réseau IP. La sécurité d'une telle traversée dépend beaucoup des méthodes d'authentification et d'encapsulation fournies par la mise en œuvre, et choisie durant la négociation entre le client et le serveur SOCKS.

Une considération attentive devrait être apportée par les administrateur au choix de la méthode d'authentification.

9. Références

[1] Koblas, D., "SOCKS", Proceedings: 1992 Usenix Security Symposium.

Adresse de l'auteur

Marcus Leech
Bell-Northern Research Ltd
P.O. Box 3511, Stn. C,
Ottawa, ON
CANADA K1Y 4H7
téléphone : (613) 763-9145
mél : mleech@bnr.ca