Groupe de travail Réseau
Request for Comments : 4254
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

T. Ylonen, SSH Communications Security Corp
C. Lonvick, éd. ,Cisco Systems, Inc.

janvier 2006

Protocole de connexion Secure Shell (SSH)

Statut de ce 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
Secure Shell (SSH) est un protocole pour la connexion à distance sécurisée et autres services réseau sécurisés sur un réseau non sûr.

Ce document décrit les protocole de connexion SSH. Il fournit la connexion interactive des sessions, l'exécution à distance des commandes, la transmission TCP/IP des connexions, et la transmission X11 des connexions. Tous ces canaux sont multiplexés en un seul tunnel chiffré.

Le protocole de connexion SSH a été conçu pour fonctionner par dessus la couche de transport SSH et les protocoles d'authentification d'utilisateur.

Table des matières

1. Introduction

Le protocole de connexion SSH a été conçu pour fonctionner par dessus la couche de transport SSH et les protocoles d'authentification d'utilisateur ([RFC4253] et [RFC4252]). Il fournit des sessions de connexion interactives, l'exécution de commandes à distance, des connexions TCP/IP transmises, et des connexions X11 transmises.

Le 'nom de service' pour ce protocole est "ssh-connection".

Le présent document devrait n'être lu qu'après le document d'architecture SSH [RFC4251]. Le présent document utilise librement la terminologie et la notation provenant du document d'architecture sans autre référence ou explication.

2. Contributeurs

Les contributeurs majeurs originaux de cet ensemble de documents ont été Tatu Ylonen, Tero Kivinen, Timo J. Rinne, Sami Lehtinen (tous de SSH Communications Security Corp) et Markku-Juhani O. Saarinen (Université de Jyvaskyla). Darren Moffat était l'éditeur original de cet ensemble de documents et y a aussi fait de très substantielles contributions.

De nombreuses personnes ont contribué au développement de ce document au fil des ans. Les personnes qui doivent en être remerciées incluent Mats Andersson, Ben Harris, Bill Sommerfeld, Brent McClure, Niels Moller, Damien Miller, Derek Fawcus, Frank Cusack, Heikki Nousiainen, Jakob Schlyter, Jeff Van Dyke, Jeffrey Altman, Jeffrey Hutzelman, Jon Bright, Joseph Galbraith, Ken Hornstein, Markus Friedl, Martin Forssen, Nicolas Williams, Niels Provos, Perry Metzger, Peter Gutmann, Simon Josefsson, Simon Tatham, Wei Dai, Denis Bider, der Mouse, et Tadayoshi Kohno. La présence de leur noms ici ne signifie pas qu'ils approuvent le présent document, mais qu'il y ont contribué.

3. Conventions utilisées dans ce document

Tous les documents relatifs aux protocoles SSH devront utiliser 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 qui sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Les mots-clés "UTILISATION PRIVÉE", "ALLOCATION HIÉRARCHIQUE ", "PREMIER ARRIVÉ, PREMIER SERVI", "REVUE D'EXPERT", "SPÉCIFICATION EXIGÉE", "APPROBATION DE L'IESG", "CONSENSUS DE L'IETF", et "ACTION DE NORMALISATION" qui apparaissent dans le présent document lorsque ils sont utilisés pour décrire l'allocation d'espace de noms sont à interpréter comme décrit dans la [RFC2434].

Les champs de protocole et les valeurs possibles pour les remplir sont définis dans cet ensemble de documents. Les champs de protocole seront définis dans les définitions de messages. Par exemple, SSH_MSG_CHANNEL_DATA *(données de canal de message SSH)* est défini comme suit :

octet : SSH_MSG_CHANNEL_DATA
uint32 *(entier non signé sur 32 bits)* : canal receveur
chaîne : données

Tout au long de ces documents, quand les champs sont référencés, ils vont apparaître avec des guillemets simples. Quand des valeurs pour remplir ces champs sont référencées, elles vont apparaître avec des guillemets doubles. En utilisant l'exemple ci-dessus, les valeurs possibles pour 'données' sont "foo" et "bar".

4. Demandes globales

Il y a plusieurs sortes de demandes qui affectent l'état de l'extrémité distante globalement, indépendamment de tous canaux. Un exemple est une demande de commencer une transmission TCP/IP pour un accès spécifique. Noter que le client et le serveur PEUVENT tous deux envoyer des demandes globales à tout moment, et le receveur DOIT répondre de façon appropriée. Toutes ces demandes utilisent le format suivant.

octet : SSH_MSG_GLOBAL_REQUEST *(message SSH de demande globale)*
chaîne : nom de la demande en US-ASCII seulement
booléen : veut une réponse
.... les données spécifiques de la demande suivent

La valeur de 'nom de demande' suit la convention de désignation d'extensibilité du DNS mentionnée dans la [RFC4251].

Le receveur va répondre à ce message avec SSH_MSG_REQUEST_SUCCESS *(message SSH de demande réussie)* ou SSH_MSG_REQUEST_FAILURE *(message SSH d'échec de demande)* si 'veut une réponse' est VRAI.

octet : SSH_MSG_REQUEST_SUCCESS
.... données spécifique de la réponse

Généralement, les 'données spécifiques de la réponse' n'existent pas

Si le receveur ne reconnaît pas ou n'accepte pas la demande, il répond simplement par SSH_MSG_REQUEST_FAILURE.

octet : SSH_MSG_REQUEST_FAILURE

En général, les messages de réponse n'incluent pas d'identifiant de type de demande. Pour permettre au générateur d'une demande d'identifier à quelle demande chaque réponse se réfère, il est EXIGÉ que les réponses à SSH_MSG_GLOBAL_REQUEST soient envoyées dans le même ordre que les messages de demande correspondants. Pour les demandes de canaux, les réponses qui se rapportent au même canal DOIVENT aussi être envoyées dans le bon ordre. Cependant, les demandes de canaux pour des canaux distincts PEUVENT être envoyées dans le désordre.

5. Mécanisme de canal

Toutes les sessions terminales, les connexions transmises, etc., sont des canaux. L'un ou l'autre côté peut ouvrir un canal. Plusieurs canaux sont multiplexés dans une seule connexion.

Les canaux sont identifié par des numéros à chaque extrémité. Le numéro qui se réfère à un canal peut être différent de chaque côté. Les demandes d'ouverture d'un canal contiennent le numéro de canal de l'envoyeur. Tous les autres messages relatifs à un canal contiennent le numéro de canal du receveur pour le canal.

Les canaux sont contrôlés par flux. Aucune donnée ne peut être envoyée à un canal avant qu'un message soit reçu pour indiquer quel espace de fenêtre est disponible.

5.1 Ouverture d'un canal

Quand l'un ou l'autre côté souhaite ouvrir un nouveau canal, il alloue un numéro local pour le canal. Il envoie ensuite le message suivant à l'autre côté, et inclut le numéro de canal local et la taille de fenêtre initiale dans le message.

octet : SSH_MSG_CHANNEL_OPEN *(message SSH canal ouvert)*
chaîne : type de canal en US-ASCII seulement
uint32 : canal d'envoi
uint32 : taille de fenêtre initiale
uint32 : taille maximum de paquet
.... les données spécifique du type de canal suivent

Le 'type de canal' est un nom, comme décrit dans la [RFC4251] et la [RFC4250], avec des mécanismes d'extension similaires. Le 'canal d'envoi' est un identifiant local pour le canal utilisé par l'envoyeur de ce message. La 'taille de fenêtre initiale' spécifie combien d'octets de données de canal peuvent être envoyées à l'envoyeur de ce message sans ajuster la fenêtre. La 'taille maximum de paquet' spécifie la taille maximum d'un paquet individuel de données qui peut être envoyé à l'envoyeur. Par exemple, on peut vouloir utiliser de plus petits paquets pour les connexions interactives pour obtenir de meilleures réponses interactives sur des liaisons lentes.

Le côté distant décide alors si il peut ouvrir le canal, et répond soit par SSH_MSG_CHANNEL_OPEN_CONFIRMATION *(message SSH de confirmation de canal ouvert)*, soit par SSH_MSG_CHANNEL_OPEN_FAILURE *(message SSH d'échec d'ouverture de canal)*.

octet : SSH_MSG_CHANNEL_OPEN_CONFIRMATION
uint32 : canal receveur
uint32 : canal d'envoi
uint32 : taille de fenêtre initiale
uint32 : taille maximum de paquet
.... les données spécifiques du type de canal suivent

Le 'canal receveur' est le numéro de canal donné dans la demande d'ouverture originale, et 'canal d'envoi' est le numéro de canal alloué par l'autre côté.

octet : SSH_MSG_CHANNEL_OPEN_FAILURE
uint32 : canal receveur
uint32 : code de cause
chaîne : description en codage ISO-10646 UTF-8 [RFC3629]
chaîne : étiquette de langage [RFC3066]

Si le receveur du message SSH_MSG_CHANNEL_OPEN ne prend pas en charge le 'type de canal' spécifié, il répond simplement par SSH_MSG_CHANNEL_OPEN_FAILURE. Le client PEUT montrer la chaîne 'description' à l'utilisateur. Si c'est fait, le logiciel de client devrait prendre les précautions discutées dans la [RFC4251].

Les valeurs de 'code de cause' de SSH_MSG_CHANNEL_OPEN_FAILURE sont définies dans le tableau qui suit. Noter que les valeurs de 'code de cause' sont données en format décimal pour qu'elles soient lisibles, mais ce sont en fait des valeurs uint32.

Nom symbolique	**Code de cause**	*(Signification)*
SSH_OPEN_ADMINISTRATIVELY_PROHIBITED	1	*(ouverture administrativement interdite)*
SSH_OPEN_CONNECT_FAILED	2	*(échec d'ouverture de connexion)*
SSH_OPEN_UNKNOWN_CHANNEL_TYPE	3	*(type de canal inconnu)*
SSH_OPEN_RESOURCE_SHORTAGE	4	*(manque de ressources pour l'ouverture)*

Les demandes d'allocation de nouvelles valeurs de code de cause de SSH_MSG_CHANNEL_OPEN (et du texte de description associé) dans la gamme de 0x00000005 à 0xFDFFFFFF DOIVENT être faites par la méthode de CONSENSUS DE L'IETF, comme décrit dans la [RFC2434]. L'IANA n'allouera pas de valeurs de code de cause d'échec de connexion de canal dans la gamme de 0xFE000000 à 0xFFFFFFFF. Les valeurs de code de cause d'échec de connexion de canal dans cette gamme soit laissées pour UTILISATION PRIVÉE, comme décrit dans la [RFC2434].

Bien qu'il soit entendu que l'IANA n'aura aucun contrôle sur la gamme de 0xFE000000 à 0xFFFFFFFF, celle-ci sera partagée en deux parties et administrée par les conventions suivantes :

- La gamme de 0xFE000000 à 0xFEFFFFFF est à utiliser en conjonction avec des canaux alloués en local. Par exemple, si un canal est proposé avec un type de canal de "exemple_session@exemple.com", mais échoue, la réponse contiendra un code de cause alloué par l'IANA (comme mentionné ci-dessus et dans la gamme de 0x00000001 à 0xFDFFFFFF) ou une valeur allouée en local dans la gamme de 0xFE000000 à 0xFEFFFFFF. Naturellement, si le serveur ne comprend pas le type de canal proposé, même si c'est un type de canal défini en local, alors le code de cause DOIT être 0x00000003, comme décrit ci-dessus, si le code de cause est envoyé. Si le serveur comprend le type de canal, mais si il échoue encore à s'ouvrir, le serveur DEVRAIT alors répondre avec un code de cause alloué en local cohérent avec le type de canal proposé en local. On suppose qu'en pratique on tentera d'abord d'utiliser les valeurs de code de cause allouées par l'IANA et qu'ensuite on tentera les valeurs de code de cause allouées en local.

- Il n'y a pas de restriction ni suggestion pour la gamme commençant à 0xFF. Aucune interopérabilité n'est attendue pour ce qui est utilisé dans cette gamme. Elle est essentiellement pour l'expérimentation.

5.2 Transfert de données

La taille de fenêtre spécifie combien d'octets l'autre partie peut envoyer avant qu'elle doive attendre pour que la fenêtre soit ajustée. Les deux parties utilisent le message suivant pour ajuster la fenêtre.

octet : SSH_MSG_CHANNEL_WINDOW_ADJUST *(message SSH d'ajustement de fenêtre de canal)*
uint32 : canal receveur
uint32 : octets à ajouter

Après avoir reçu ce message, le receveur PEUT envoyer un nombre d'octets supérieur à ce qu'il était précédemment permis d'envoyer ; la taille de fenêtre est incrémentée. Les mises en œuvre DOIVENT traiter correctement les tailles de fenêtre jusqu'à 2^32 - 1 octets. La fenêtre NE DOIT PAS être augmentée au delà de 2^32 - 1 octets.

Le transfert de données est fait avec des messages du type suivant :

octet : SSH_MSG_CHANNEL_DATA
uint32 : canal receveur
chaîne : données

La quantité maximum de données permise est déterminée par la taille maximum de paquet pour le canal, et la taille de fenêtre actuelle, selon celle qui est la plus petite. La taille de fenêtre est décrémentée de la longueur des données envoyées, champ de longueur inclus. Les deux parties PEUVENT ignorer toutes les données supplémentaires envoyées après que la fenêtre permise est vide.

Les mises en œuvre sont supposées avoir une limite à la taille de paquet SSH à la couche de transport (toute limite pour les paquets reçus DOIT être de 32 768 octets ou plus, comme décrit dans la [RFC4253]). La mise en œuvre de la couche de connexion SSH :

- NE DOIT PAS annoncer une taille maximum de paquet qui résulterait en transport de paquets plus grands que ce que sa couche de transport accepte de recevoir,
- NE DOIT PAS générer de paquets de données plus grands que ce que sa couche de transport accepte d'envoyer, même si l'extrémité distante veut accepter de très gros paquets.

De plus, certains canaux peuvent transférer plusieurs types de données. Un exemple de cela sont les données stderr provenant de sessions interactives. De telles données peuvent être passées avec des messages SSH_MSG_CHANNEL_EXTENDED_DATA *(message SSH de données étendues de canal)* où un entier séparé spécifie le type des données. Les types disponibles et leur interprétation dépendent du type de canal.

octet : SSH_MSG_CHANNEL_EXTENDED_DATA
uint32 : canal receveur
uint32 : data_type_code
chaîne : données

Les données envoyées avec ces messages consomment la même fenêtre que les données ordinaires.

Actuellement, seul le type suivant est défini. Noter que la valeur pour le code de type de données est fournie en format décimal pour la lisibilité, mais les valeurs sont en fait en uint32.

Nom symbolique	**code de type de données**
SSH_EXTENDED_DATA_STDERR	1

Les valeurs de code de type de données de transfert de données de canal étendu DOIVENT être allouées en séquence. Les demandes d'allocation de nouvelles valeurs de code de type de données de transfert de données de canal étendu et leurs chaînes associées de données de transfert de données de canal étendu, dans la gamme de 0x00000002 à 0xFDFFFFFF, DOIVENT être faites par la méthode de CONSENSUS DE L'IETF comme décrit dans la [RFC2434]. L'IANA n'allouera pas de valeurs de code de type de données de transfert de données de canal étendu dans la gamme de 0xFE000000 à 0xFFFFFFFF. Les valeurs de code de type de données de transfert de données de canal étendu dans cette gamme sont laissées pour UTILISATION PRIVÉE, comme décrit dans la [RFC2434]. Comme on l'a noté, les instructions réelles à l'IANA sont dans la [RFC4250].

5.3 Fermeture d'un canal

Quand une partie ne va plus envoyer d'autres données à un canal, elle DEVRAIT envoyer SSH_MSG_CHANNEL_EOF *(fin de fichier de canal de message SSH)*.

octet : SSH_MSG_CHANNEL_EOF
uint32 : canal receveur

Aucune réponse explicite n'est envoyée à ce message. Cependant, l'application peut envoyer un EOF *(fin de fichier)* à ce qui se trouve à l'autre extrémité du canal. Noter que le canal reste ouvert après ce message, et plus de données peuvent encore être envoyées dans l'autre direction. Ce message ne consomme pas d'espace de fenêtre et peut être envoyé même si aucun espace de fenêtre n'est disponible.

Quand l'une ou l'autre partie souhaite terminer le canal, elle envoie SSH_MSG_CHANNEL_CLOSE *(message SSH de clôture de canal)*. À réception de ce message, une partie DOIT renvoyer un SSH_MSG_CHANNEL_CLOSE sauf si il a déjà envoyé ce message pour le canal. Le canal est considéré fermé pour une partie quand elle a envoyé et reçu le SSH_MSG_CHANNEL_CLOSE, et cette partie peut alors réutiliser le numéro de canal. Une partie PEUT envoyer SSH_MSG_CHANNEL_CLOSE sans avoir envoyé ou reçu de SSH_MSG_CHANNEL_EOF.

octet : SSH_MSG_CHANNEL_CLOSE
uint32 ; canal receveur

Ce message ne consomme pas d'espace de fenêtre et peut être envoyé même si aucun espace de fenêtre n'est disponible.

Il est RECOMMANDÉ que toutes les données envoyées avant ce message soient livrées à la destination réelle, si possible.

5.4 Demandes spécifiques de canal

De nombreuses valeurs de 'type de canal' ont des extensions qui sont spécifiques de ce type de canal particulier. Un exemple est celui de la demande d'un pty (pseudo terminal) pour une session interactive.

Toutes les demandes spécifiques de canal utilisent le format suivant :

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : type de demande seulement en caractères US-ASCII
booléen : veut une réponse
.... les données spécifiques du type de canal suivent

Si 'veut une réponse' est FAUX, aucune réponse ne sera envoyée à la demande. Autrement, le receveur répond soit par SSH_MSG_CHANNEL_SUCCESS, SSH_MSG_CHANNEL_FAILURE, soi par des messages de continuation spécifiques de la demande. Si la demande n'est pas reconnue ou n'est pas prise en charge pour le canal, SSH_MSG_CHANNEL_FAILURE est retourné.

Ce message ne consomme pas d'espace de fenêtre et peut être envoyé même si aucun espace de fenêtre n'est disponible. Les valeurs de 'type de demande' sont locales pour chaque type de canal.

Il est permis au client d'envoyer d'autres messages sans attendre la réponse à la demande.

Les noms de 'type de demande' suivent la convention de désignation d'extensibilité du DNS mentionnée dans les [RFC4251] et [RFC4250].

octet : SSH_MSG_CHANNEL_SUCCESS
uint32 : canal receveur
octet : SSH_MSG_CHANNEL_FAILURE
uint32 : canal receveur

Ces messages ne consomment pas d'espace de fenêtre et peuvent être envoyés même si aucun espace de fenêtre n'est disponible.

6. Sessions interactives

Une session est une exécution distante d'un programme. Le programme peut être une coquille, une application, une commande de système, ou un sous système incorporé. Elle peut ou non avoir un pty, et peut ou non impliquer une transmission X11. Plusieurs sessions peuvent être actives simultanément.

6.1 Ouverture d'une session

Une session commence par l'envoi du message suivant :

octet : SSH_MSG_CHANNEL_OPEN
chaîne : "session"

uint32 : canal d'envoi
uint32 : taille de fenêtre initiale
uint32 : taille maximum de paquet

Les mises en œuvre de client DEVRAIENT rejeter toutes les demandes d'ouverture de canal de session pour rendre plus difficile à un serveur corrompu d'attaquer le client.

6.2 Demande de pseudo terminal

Un pseudo terminal peut être alloué pour la session par l'envoi du message suivant :

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "pty-req"
booléen : veut une réponse
chaîne : valeur de variable d'environnement TERM (par exemple, vt100)
uint32 : largeur de terminal , caractères (par exemple, 80)
uint32 : hauteur de terminal, rangées (par exemple, 24)
uint32 ; largeur de terminal, pixels (par exemple, 640)
uint32 : hauteur de terminal, pixels (par exemple, 480)
chaîne : modes de terminal codés

Les modes de terminal codés sont décrits à la Section 8. Les paramètres de dimension zéro DOIVENT être ignorés. Les dimensions de caractère/rangée outrepassent les dimensions en pixels (quand elles sont différentes de zéro). Les dimensions en pixels se réfèrent à la zone traçable de la fenêtre.

Les paramètres de dimension sont seulement pour information.

Le client DEVRAIT ignorer les demandes de pty.

6.3 Transmission X11

6.3.1 Demande de transmission X11

La transmission X11 peut être demandée pour une session par l'envoi d'un message SSH_MSG_CHANNEL_REQUEST.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "x11-req"
booléen : veut une réponse
booléen : une seule connexion
chaîne : protocole d'authentification x11
chaîne : mouchard d'authentification x11
uint32 : numéro d'écran x11

Il est RECOMMANDÉ que le 'mouchard d'authentification x11' qui est envoyé soit un faux mouchard aléatoire,et que le mouchard soit vérifié et remplacé par le mouchard réel quand une demande de connexion est reçue.

La transmission de connexion x11 devrait s'arrêter quand le canal de session est fermé. Cependant, les transmissions déjà ouvertes ne devraient pas être automatiquement closes quand le canal de session est fermé.

Si 'une seule connexion' est VRAI, une seule connexion devrait être transmise. Aucune autre connexion ne sera transmise après la première, ou après que le canal de session a été clos.

Le 'protocole d'authentification x11' est le nom de la méthode d'authentification X11 utilisée, par exemple, "MIT-MAGIC-COOKIE-1".

Le 'mouchard d'authentification x11' DOIT être codé en hexadécimal.

Le protocole X est documenté dans [SCHEIFLER].

6.3.2 Canaux X11

Les canaux X11 sont ouverts avec une demande d'ouverture de canal. Les canaux résultants sont indépendants de la session, et la clôture du canal de session ne clôt pas les canaux X11 transmis.

octet : SSH_MSG_CHANNEL_OPEN
chaîne : "x11"
uint32 : canal d'envoi
uint32 : taille de fenêtre initiale
uint32 : taille maximum de paquet
chaîne : adresse du générateur (par exemple, "192.168.7.38")
uint32 : accès du générateur

Le receveur devrait répondre par SSH_MSG_CHANNEL_OPEN_CONFIRMATION ou SSH_MSG_CHANNEL_OPEN_FAILURE.

Les mises en œuvre DOIVENT rejeter toutes les demandes d'ouverture de canal X11 si elles n'ont pas demandé une transmission X11.

6.4 Passage des variables d'environnement

Les variables d'environnement peuvent être passées à la coquille/commande pour commencer plus tard. Le réglage non contrôlé des variables d'environnement dans un processus privilégié peut faire courir des risques pour la sécurité. Il est recommandé que les mises en œuvre tiennent une liste des noms de variables admis ou qu'elles ne règlent les variables d'environnement qu'après que le processus serveur a donné des privilèges suffisants.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "env"
booléen : veut une réponse
chaîne : nom de variable
chaîne : valeur de variable

6.5 Début d'une coquille ou d'une commande

Une fois que la session a été établie, un programme est lancé à l'extrémité distante. Le programme peut être une coquille, un programme d'application, ou un sous système avec un nom indépendant de l'hôte. Une seule de ces demandes peut réussir par canal.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "shell"
booléen : veut une réponse

Ce message va demander que la coquille par défaut de l'utilisateur (normalement définie dans /etc/passwd dans les systèmes UNIX) soit commencée à l'autre extrémité.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "exec"
booléen : veut une réponse
chaîne : commande

Ce message va demander que le serveur commence l'exécution de la commande donnée. La chaîne 'commande' peut contenir un chemin. Les précautions normales DOIVENT être prises pour empêcher l'exécution de commandes non autorisées.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "sous système"
booléen : veut une réponse
chaîne : nom de sous système

Cette dernière forme exécute un sous système prédéfini. On s'attend à ce que cela inclue un mécanisme général de transfert de fichier, et éventuellement d'autres caractéristiques. Les mises en œuvre peuvent aussi permettre de configurer plusieurs de ces mécanismes. Comme la coquille de l'utilisateur est généralement utilisée pour exécuter le sous système, il est conseillé que le protocole de sous système ait un "mouchard magique" au début de la transaction de protocole pour la distinguer d'un résultat arbitraire généré par les scripts d'initialisation de coquille, etc. Ce résultat parasite provenant de la coquille peut être filtré soit chez le serveur, soit chez le client.

Le serveur NE DEVRAIT PAS arrêter l'exécution de la pile de protocole quand il débute une coquille ou un programme. Toutes les entrées et résultats DEVRAIENT être redirigés sur le canal ou sur le tunnel chiffré.

Il est RECOMMANDÉ que la réponse à ces messages soit demandée et vérifiée. Le client DEVRAIT ignorer ces messages.

Les noms de sous systèmes suivent la convention de désignation d'extensibilité du DNS mentionnées dans la [RFC4250].

6.6 Transfert des données de session

Le transfert des données pour une session est fait en utilisant les paquets SSH_MSG_CHANNEL_DATA et SSH_MSG_CHANNEL_EXTENDED_DATA et le mécanisme de fenêtre. Le type de données étendu SSH_EXTENDED_DATA_STDERR a été défini pour les données stderr.

6.7 Message de changement de dimension de fenêtre

Quand la taille de fenêtre (terminal) change du côté du client, il PEUT envoyer un message à l'autre côté pour l'informer des nouvelles dimensions.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "changement de fenêtre"
booléen : FAUX
uint32 : largeur de terminal, colonnes
uint32 : hauteur de terminal, rangées
uint32 : largeur de terminal, pixels
uint32 : hauteur de terminal, pixels

Une réponse NE DEVRAIT PAS être envoyée à ce message.

6.8 Contrôle de flux local

Sur de nombreux systèmes, il est possible de déterminer si un pseudo terminal utilise le contrôle de flux "control-S/control-Q". Quand le contrôle de flux est permis, il est souvent souhaitable de le faire à l'extrémité client pour accélérer les réponses aux demandes de l'utilisateur. Ceci est facilité par la notification suivante. Initialement, le serveur est responsable du contrôle de flux. (Ici encore, client signifie le côté à l'origine de la session, et serveur signifie l'autre côté.)

Le message ci-dessous est utilisé par le serveur pour informer le client quand il peut ou non effectuer le contrôle de flux (traitement control-S/control-Q). Si 'client peut faire' est VRAI, il est permis au client de faire le contrôle de flux en utilisant control-S et control-Q. Le client PEUT ignorer ce message.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "xon-xoff"
booléen : FAUX
booléen : client peut faire

Aucune réponse n'est envoyée à ce message.

6.9 Signaux

Un signal peut être délivré au processus/service distant en utilisant le message suivant. Certains systèmes peuvent ne pas mettre en œuvre les signaux, et dans ce cas, ils DEVRAIENT ignorer ce message.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "signal"
booléen : FAUX
chaîne : nom du signal (sans le préfixe "SIG")

Les valeurs de 'nom de signal' seront codées comme discuté dans le passage qui décrit les messages SSH_MSG_CHANNEL_REQUEST qui utilisent "exit-signal" dans cette section.

6.10 Retour de l'état de sortie

Lorsque la commande exécutée à l'autre extrémité se termine, le message suivant peut être envoyé pour communiquer l'état de sortie de la commande. Retourner l'état est RECOMMANDÉ. Aucun accusé de réception n'est envoyé pour ce message. Le canal doit être clos par le serveur avec SSH_MSG_CHANNEL_CLOSE après ce message.

Le client PEUT ignorer ces messages.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "état de sortie"
booléen : FAUX
uint32 : état de sortie

La commande distante peut aussi se terminer violemment à cause d'un signal. Une telle condition peut être indiquée par le message suivant. Un état de sortie de zéro veut généralement dire que la commande s'est terminée avec succès.

octet : SSH_MSG_CHANNEL_REQUEST
uint32 : canal receveur
chaîne : "signal de sortie"
booléen : FAUX
chaîne : nom de signal (sans le préfixe "SIG")
booléen : core dumped *(vidage du cœur)*
chaîne : le message d'erreur en codage ISO-10646 UTF-8
chaîne : étiquette de langue [RFC3066]

Le 'nom de signal' est un des suivants (tirés de [POSIX]) :

ABRT
ALRM
FPE
HUP
ILL
INT
KILL
PIPE
QUIT
SEGV
TERM
USR1
USR2

Des valeurs supplémentaires de nom de signal PEUVENT être envoyées dans le format "nom-de-signal@xyz", où "nom-de-signal" et "xyz" peuvent être tout ce que veut une mise en œuvre particulière (sauf le signe "@"). Cependant, il est suggéré que si un script 'configure' est utilisé, toute valeur non standard de 'nom de signal' qu'elle trouve soit codée comme "SIG@xyz.config.guess", où "SIG" est le 'nom de signal' sans le préfixe "SIG", et "xyz" est le type d'hôte, comme déterminé par "config.guess".

Le 'message d'erreur' contient une explication textuelle supplémentaire du message d'erreur. Le message peut consister en plusieurs lignes séparées par des paires de CRLF (Retour chariot - saut à la ligne). Le logiciel de client PEUT afficher ce message à l'utilisateur. Si c'est fait, le logiciel client devrait prendre les précautions discutées dans la [RFC4251].

7. Transmission d'accès TCP/IP

7.1 Demande de transmission d'accès

Une partie n'a pas besoin de demander explicitement des transmission à partir de sa propre extrémité vers l'autre direction. Cependant, si elle souhaite que les connexions à un accès sur l'autre côté soient transmises au côté local, elle doit le demander explicitement.

octet : SSH_MSG_GLOBAL_REQUEST
chaîne : "tcpip-forward"
booléen : veut une réponse
chaîne : adresse à lier (par exemple, "0.0.0.0")
uint32 : numéro d'accès à lier

L'adresse à lier et le numéro d'accès à lier spécifient l'adresse IP (ou le nom de domaine) et l'accès sur lequel les connexions pour transmission doivent être acceptées. Certaines chaînes utilisées pour 'adresse à lier' ont une sémantique de casse particulière :

- "" signifie que les connexions doivent être acceptées sur toutes les familles de protocole prises en charge par la mise en œuvre de SSH.
- "0.0.0.0" signifie d'écouter sur toutes les adresses IPv4.
- "::" signifie d'écouter sur toutes les adresses IPv6.
- "localhost" signifie d'écouter sur toutes les familles de protocole prises en charge par la mise en œuvre SSH seulement sur les adresses de rebouclage ([RFC3330] et [RFC3513]).
- "127.0.0.1" et "::1" indiquent d'écouter sur les interfaces de bouclage pour, respectivement, IPv4 et IPv6.

Noter que le client peut toujours filtrer les connexions sur la base des informations passées dans la demande d'ouverture.

Les mises en œuvre devraient ne permettre la transmission que sur les accès privilégiés si l'utilisateur a été authentifié comme utilisateur privilégié.

Les mises en œuvre de client DEVRAIENT rejeter ces messages ; il ne sont normalement envoyés que par le client.

Si un client passe 0 comme numéro d'accès à lier et a 'veut une réponse' à VRAI, alors le serveur alloue le prochain numéro d'accès disponible non privilégié et répond avec le message suivant ; autrement, il n'y a pas de données spécifiques de la réponse.

octet : SSH_MSG_REQUEST_SUCCESS
uint32 : accès qui était lié sur le serveur

Une transmission d'accès peut être annulée avec le message suivant. Noter que les demandes d'ouverture de canal peuvent être reçues jusqu'à ce qu'une réponse à ce message soit reçue.

octet : SSH_MSG_GLOBAL_REQUEST
chaîne : "cancel-tcpip-forward"
booléen : veut une réponse
chaîne : adresse à lier (par exemple, "127.0.0.1")
uint32 : numéro d'accès à lier

Les mises en œuvre de client DEVRAIENT rejeter ces messages ; ils ne sont normalement envoyés que par le client.

7.2. Canaux de transmission TCP/IP

Quand une connexion arrive à un accès pour lequel la transmission à distance a été demandée, un canal est ouvert pour transmettre l'accès à l'autre côté.

octet : SSH_MSG_CHANNEL_OPEN
chaîne : "forwarded-tcpip"
uint32 : canal d'envoi
uint32 : taille de fenêtre initiale
uint32 : taille maximum de paquet
chaîne : adresse connectée
uint32 : accès connecté

chaîne : adresse IP de l'origine
uint32 : accès de l'origine

Les mises en œuvre DOIVENT rejeter ces messages sauf si elles avaient précédemment demandé une transmission d'accès TCP/IP distante avec ce numéro d'accès.

Quand une connexion arrive à un accès TCP/IP transmis localement, le paquet suivant est envoyé à l'autre côté. Noter que ces messages PEUVENT aussi être envoyés pour des accès pour lesquels aucune transmission n'avait été explicitement demandée. Le côté receveur doit décider si il permet la transmission.

octet : SSH_MSG_CHANNEL_OPEN
chaîne : "direct-tcpip"
uint32 : canal d'envoi
uint32 : taille de fenêtre initiale
uint32 : taille maximum de paquet
chaîne : hôte à connecter
uint32 : accès à connecter
chaîne : adresse IP de l'origine
uint32 : accès de l'origine

Le 'hôte à connecter' et le 'accès à connecter' spécifient l'hôte et l'accès TCP/IP où le receveur devrait connecter le canal. Le 'hôte à connecter' peut être soit un nom de domaine, soit une adresse IP numérique.

Le 'adresse IP de l'origine' est l'adresse IP numérique de la machine d'où provient la demande de connexion, et le 'accès de l'origine' est l'accès sur l'hôte d'où provient la connexion.

Les canaux TCP/IP transmis sont indépendants de toutes les sessions, et la clôture d'un canal de session n'implique en aucune façon que les connexions transmises devraient être fermées.

Les mises en œuvre de client DEVRAIENT rejeter les demandes directes d'ouverture TCP/IP pour des raisons de sécurité.

8. Codage des modes de terminal

Tous les 'modes de terminal codés' (comme passés dans une demande de pty) sont codés en flux d'octets. L'intention est que le codage soit portable à travers des environnements différents. Le flux consiste en paires de opcode-argument dans lesquels le opcode est une valeur d'octet. Les opcodes 1 à 159 ont un seul argument uint32. Les opcodes 160 à 255 ne sont pas encore définis, et causent l'arrêt de l'analyse (ils ne devraient être utilisés qu'après d'autres données). Le flux se termine par l'opcode TTY_OP_END (0x00).

Le client DEVRAIT mettre tous les modes qu'il connaît dans le flux, et le serveur PEUT ignorer tout mode qu'il ne connaît pas. Cela permet un certain degré d'indépendance de la machine, au moins entre les systèmes qui utilisent une interface de style POSIX. Le protocole peut aussi prendre en charge d'autres systèmes, mais le client peut avoir besoin de mettre des valeurs raisonnables pour un certain nombre de paramètres afin que le pty de serveur soit réglé à un mode raisonnable (le serveur laisse tous les bits de mode non spécifiés à leur valeur par défaut, et seules certaines combinaisons ont un sens).

La désignation des valeurs de opcode suit principalement les fanions de mode terminal POSIX. Les valeurs de opcode suivantes ont été définies. Noter que les valeurs données ci-dessous sont en format décimal pour la lisibilité, mais ce sont en fait des valeurs d'octets.

Opcode	Mnémonique	Description
0	TTY_OP_END	Indique la fin des options.
1	VINTR	Caractère d'interruption ; 255 si aucun. De même pour les autres caractères. Tous ces caractères ne sont pas supportés sur tous les systèmes.
2	VQUIT	Caractère quitte (envoie le signal SIGQUIT sur les systèmes POSIX).
3	VERASE	Écrase le caractère à gauche du curseur.
4	VKILL	Écrase la ligne d'entrée en cours.
5	VEOF	Caractère fin de fichier (envoie EOF à partir du terminal).

6	VEOL	Caractère fin de ligne en plus de retour chariot et/ou saut à la ligne.
7	VEOL2	Caractère fin de ligne supplémentaire.
8	VSTART	Continue la pause de résultat (normalement contrôle-Q).
9	VSTOP	Pause du résultat (normalement contrôle-S).
10	VSUSP	Suspend le programme en cours.
11	VDSUSP	Autre caractère de suspension.
12	VREPRINT	Ré imprime la ligne d'entrée en cours.
13	VWERASE	Écrase un mot à gauche du curseur.
14	VLNEXT	Entre le prochain caractère tapé littéralement, même si c'est un caractère spécial
15	VFLUSH	Caractère pour purger le résultat.
16	VSWTCH	Passe à une couche de coquille différente.
17	VSTATUS	Imprime la ligne d'état de système (charge, commande, pid, etc).
18	VDISCARD	Bascule la purge d'un résultat terminal.
30	IGNPAR	Fanion ignorer la parité. Le paramètre DEVRAIT être 0 si ce fanion est FAUX, et 1 si il est VRAI.
31	PARMRK	Erreurs de parité et de tramage de marque.
32	INPCK	Permet la vérification d'erreurs de parité.
33	ISTRIP	Supprime le huitième bit des caractères.
34	INLCR	Transpose NL en CR sur l'entrée.
35	IGNCR	Ignorer le CR en entrée.
36	ICRNL	Transpose CR en NL sur l'entrée.
37	IUCLC	Transpose les caractères majuscules en minuscules.
38	IXON	Active le contrôle de flux en sortie.
39	IXANY	Tout caractère va recommencer après l'arrêt.
40	IXOFF	Active le contrôle de flux en entrée.
41	IMAXBEL	Sonnerie quand la file d'attente d'entrée est pleine.
50	ISIG	Active les signaux INTR, QUIT, [D]SUSP.
51	ICANON	Canonise les lignes d'entrée.
52	XCASE	Permet l'entrée et la sortie de caractères majuscules en faisant précéder leur équivalent minuscule de "\".
53	ECHO	Active l'écho.
54	ECHOE	Visualisation des caractères écrasés.
55	ECHOK	Le caractère Kill élimine la ligne en cours.
56	ECHONL	Écho de NL même si ECHO est désactivé.
57	NOFLSH	Ne pas purger après l'interruption.
58	TOSTOP	Arrêt des tâches d'arrière plan provenant de la sortie.
59	IEXTEN	Active les extensions.
60	ECHOCTL	Faire écho des caractères de contrôle avec ^(Char).
61	ECHOKE	Écrasement visuel pour suppression de ligne.
62	PENDIN	Refrappe de l'entrée en cours.
70	OPOST	Permet le traitement de la sortie.
71	OLCUC	Convertit les minuscules en majuscules.
72	ONLCR	Transpose NL en CR-NL.

73	OCRNL	Traduit le retour chariot en nouvelle ligne (en sortie).
74	ONOCR	Traduit la nouvelle ligne en retour chariot-nouvelle ligne (en sortie).
75	ONLRET	Nouvelle ligne effectue un retour chariot (en sortie).
90	CS7	Mode 7 bits.
91	CS8	Mode 8 bits.
92	PARENB	Parité activée.
93	PARODD	Parité impaire, autrement, paire.
128	TTY_OP_ISPEED	Spécifie le taux d'entrée en bauds en bits par seconde.
129	TTY_OP_OSPEED	Spécifie le taux de sortie en bauds en bits par seconde.

9. Résumé des numéros de message

Voici un sommaire des messages avec le numéro de message associé :

SSH_MSG_GLOBAL_REQUEST	80
SSH_MSG_REQUEST_SUCCESS	81
SSH_MSG_REQUEST_FAILURE	82
SSH_MSG_CHANNEL_OPEN	90
SSH_MSG_CHANNEL_OPEN_CONFIRMATION	91
SSH_MSG_CHANNEL_OPEN_FAILURE	92
SSH_MSG_CHANNEL_WINDOW_ADJUST	93
SSH_MSG_CHANNEL_DATA	94
SSH_MSG_CHANNEL_EXTENDED_DATA	95
SSH_MSG_CHANNEL_EOF	96
SSH_MSG_CHANNEL_CLOSE	97
SSH_MSG_CHANNEL_REQUEST	98
SSH_MSG_CHANNEL_SUCCESS	99
SSH_MSG_CHANNEL_FAILURE	100

10. Considérations relatives à l'IANA

Ce document fait partie d'un ensemble. Les considérations relatives à l'IANA pour le protocole SSH telles que définies dans les [RFC4251], [RFC4252], [RFC4253], et le présent document, sont détaillées dans la [RFC4250].

11. Considérations sur la sécurité

Ce protocole est supposé fonctionner sur un transport sûr authentifié. L'authentification de l'utilisateur et la protection contre les attaques de niveau réseau sont supposées être fournies par les protocoles sous-jacents.

Les considérations de sécurité complètes pour ce protocole sont fournies dans la [RFC4251]. Comme spécificité au présent document, il est RECOMMANDÉ que les mises en œuvre désactivent toutes les caractéristiques potentiellement dangereuses (par exemple, la transmission d'agent, la transmission X11, et la transmission TCP/IP) si la clé d'hôte a changé sans avertissement ou explication.

12. Références

12.1 Références normatives

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

[RFC2434] T. Narten et H. Alvestrand, "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC", BCP 26, octobre 1998. *(Rendue obsolète par la* RFC5226)

[RFC3066] H. Alvestrand, "Étiquettes pour l'identification des langues", BCP 47, janvier 2001. *(Obsolète, voir la* RFC4646.)

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

[RFC4250] S. Lehtinen et C. Lonvick, éd., "Numéros alloués du protocole Secure Shell (SSH)", janvier 2006. *(P.S. ; MàJ par* RFC8268)

[RFC4251] T. Ylonen et C. Lonvick, "Architecture du protocole Secure Shell (SSH)", janvier 2006. *(P.S. ; MàJ par* RFC8308)

[RFC4252] T. Ylonen et C. Lonvick, éd., "Protocole d'authentification Secure Shell (SSH)", janvier 2006. *(P.S .; MàJ par* RFC8308, 8332)

[RFC4253] C. Lonvick, "Protocole de couche Transport Secure Shell (SSH)", janvier 2006. *(P.S., MàJ par* RFC6668, 8268, 8308, 8332, 8709)

12.2 Références pour information

[POSIX] ISO/IEC, 9945-1., "Information technology -- Portable Operating System Interface (POSIX)-Part 1: System Application Program Interface (API) C Language", ANSI/IEE Std 1003.1, juillet 1996.

[RFC3330] IANA, "Adresses IPv4 d'usage particulier", septembre 2002. *(Information ; remplacée par* RFC5735)

[RFC3513] R. Hinden et S. Deering, "Architecture d'adressage du protocole Internet version 6 (IPv6)", avril 2003. (*Obs. voir* RFC4291)

[SCHEIFLER] Scheifler, R., "X Window System : The Complete Reference to Xlib, X Protocol, Icccm, Xlfd, 3rd edition.", Digital Press ISBN 1555580882, février 1992.

Adresse des auteurs

Tatu Ylonen
SSH Communications Security Corp
Valimotie 17
00380 Helsinki
Finland
mél : ylo@ssh.com

Chris Lonvick (editor)
Cisco Systems, Inc.
12515 Research Blvd.
Austin 78759
USA
mél : clonvick@cisco.com

Notice de marque commerciale

"ssh" est une marque commerciale déposée au États Unis d'Amérique et/ou dans d'autres pays.

Déclaration complète de droits de reproduction

garantie que l'utilisation des informations ci encloses ne violent aucun droit ou aucune garantie implicite de commercialisation ou d'aptitude à un objet particulier.

Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourraient être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr .

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org .

Remerciement

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