Groupe de travail Réseau
Request for Comments : 4251
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

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

janvier 2006

Architecture du protocole Secure Shell (SSH)

Résumé
Le protocole Secure Shell (SSH) est un protocole pour la connexion à distance sécurisée et autres services réseau sécurisés sur un réseau non sûr. Le présent document décrit l'architecture du protocole SSH, ainsi que la notation et la terminologie utilisées dans les documents de protocole SSH. Il discute aussi du système de dénomination des algorithmes SSH qui permettent des extensions locales. Le protocole SSH consiste en trois composants majeurs. Le protocole de couche transport assure l'authentification, la confidentialité, et l'intégrité du serveur avec le secret parfait vers l'avant. Le protocole d'authentification d'utilisateur authentifie le client auprès du serveur. Le protocole de connexion multiplexe le tunnel chiffré en sept canaux logiques. Les détails de ces protocoles sont décrits dans des documents séparés.

Table des matières

1. Introduction

Secure Shell (SSH) est un protocole pour la connexion à distance sécurisée et autres services réseau sécurisés sur un réseau non sûr. Il consiste en trois composants majeurs :

- Le protocole de couche transport [RFC4253] fournit l'authentification, la confidentialité, et l'intégrité du serveur. Il peut facultativement fournir aussi la compression. La couche transport va normalement fonctionner sur une connexion TCP/IP, mais peut aussi être utilisée par dessus tout autre flux de données fiable.

- Le protocole d'authentification d'utilisateur [RFC4252] authentifie l'utilisateur côté client auprès du serveur. Il fonctionne sur le protocole de couche transport.

- Le protocole de connexion [RFC4254] multiplexe le tunnel chiffré en sept canaux logiques. Il fonctionne sur le protocole d'authentification d'utilisateur.

Le client envoie une demande de service une fois qu'une connexion sûre de couche transport a été établie. Une seconde demande de service est envoyée après l'achèvement de l'authentification de l'utilisateur. Cela permet que de nouveaux protocoles soient définis et coexistent avec les protocoles cités ci-dessus.

Le protocole de connexion fournit des canaux qui peuvent être utilisés pour une large gamme d'objets. Des méthodes standard sont fournies pour établir des sessions dans une coquille interactive sécurisée et pour transmettre ("tunneler") des accès TCP/IP et des connexions X11 arbitraires.

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 nom 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 : canal receveur
chaîne : données

Tout au long de ces documents, quand les champs sont référencés, ils vont apparaitre 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. Architecture

4.1 Clés d'hôte

Chaque hôte serveur DEVRAIT avoir une clé d'hôte. Les hôtes PEUVENT avoir plusieurs clés d'hôte en utilisant plusieurs algorithmes différents. Plusieurs hôtes PEUVENT partager la même clé d'hôte. Si un hôte n'a pas de clé du tout, il DOIT avoir au moins une clé qui utilise chaque algorithme de clé publique EXIGÉ (DSS [FIPS-186-2]).

La clé d'hôte serveur est utilisée durant l'échange de clés pour vérifier que le client parle réellement au serveur correct. Pour que ce soit possible, le client doit avoir une connaissance a priori de la clé publique d'hôte du serveur.

Deux modèles de confiance différents peuvent être utilisés :

- Le client a une base de données locales qui associe chaque nom d'hôte (comme entré par l'utilisateur) à la clé publique d'hôte correspondante. Cette méthode n'exige pas d'infrastructure administrée centralement, et aucune coordination par un tiers. L'inconvénient est que la base de données d'associations de nom à clé peut devenir lourde à gérer.

- L'association du nom d'hôte à une clé est certifiée par une autorité de certification (CA, *certification authority*) de confiance. Le client connaît seulement la clé racine de CA, et peut vérifier la validité de toutes les clés d'hôte certifiées par les CA acceptées.

Le second modèle rend plus facile le problème de la maintenance, car idéalement une seule clé de CA doit être mémorisée de façon sûre chez le client. Par ailleurs, chaque clé d'hôte doit être certifiée de façon appropriée par une autorité centrale avant que l'autorisation soit possible. Une grosse quantité de confiance est placée dans l'infrastructure centrale.

Le protocole donne l'option que l'association nom de serveur - clé d'hôte ne soit pas vérifiée lors de la connexion à l'hôte pour la première fois. Cela permet une communication sans communication préalable des clés ou certificats d'hôte. La connexion fournit quand même la protection contre l'écoute passive ; cependant, elle devient vulnérable aux attaques par interposition actives. Les mises en œuvre NE DEVRAIENT PAS permettre normalement de telles connexions par défaut, car elles posent un problème de sécurité potentiel. Cependant, comme il n'y a pas, au moment de la rédaction du présent document, d'infrastructure de clé largement disponible sur l'Internet, cette option rend le protocole plus utilisable durant la période de transition jusqu'à ce qu'émerge une telle infrastructure, tout en fournissant quand même un plus haut niveau de sécurité que celui offert par les solutions plus anciennes (par exemple, telnet [RFC0854] et rlogin [RFC1282]).

Les mises en œuvre DEVRAIENT essayer de faire de leur mieux pour vérifier les clés d'hôte. Un exemple de stratégie possible est de n'accepter une clé d'hôte sans vérification que la première fois qu'un hôte est connecté, de sauvegarder la clé dans une base de données locale, et de comparer à cette clé toutes les connexions ultérieures à cet hôte.

Les mises en œuvre PEUVENT fournir des méthodes supplémentaires pour vérifier que les clés d'hôte sont correctes, par exemple, une empreinte digitale hexadécimale déduite du hachage SHA-1 [FIPS-180-2] de la clé publique. De telles empreintes digitales peuvent être aisément vérifiées en utilisant des canaux de communication téléphonique externes ou autres.

Toutes les mises en œuvre DEVRAIENT donner l'option de ne pas accepter les clés d'hôte qui ne peuvent pas être vérifiées.

Les membres du groupe de travail SSH estiment que la "facilité d'utilisation" est critique pour l'acceptation par l'utilisateur final des solutions de sécurité, et qu'aucune amélioration de la sécurité n'est obtenue si les nouvelles solutions ne sont pas utilisées. Donc, donner l'option de ne pas vérifier la clé d'hôte serveur est estimé améliorer la sécurité globale de l'Internet, même si cela réduit la sécurité du protocole dans les configurations où c'est permis.

4.2 Extensibilité

On estime que le protocole évoluera au fil du temps, et que certaines organisations voudront utiliser leur propres méthodes de chiffrement, d'authentification, et/ou d'échange de clés. L'enregistrement central de toutes les extensions est malcommode, en particulier pour les caractéristiques expérimentales ou classifiées. Par ailleurs, ne pas avoir d'enregistrement central conduit à des conflits d'identifiants de méthode, rendant l'interopérabilité difficile.

Nous avons choisi d'identifier les algorithmes, les méthodes, les formats, et les extensions de protocole avec des noms textuels qui sont d'un format spécifique. Les noms du DNS sont utilisés pour créer les espaces de noms locaux où les extensions expérimentales ou classifiées peuvent être définies sans crainte de conflits avec les autres mises en œuvre.

Un objectif de conception a été de garder le protocole de base aussi simple que possible, et d'exiger aussi peu que possible d'algorithmes. Cependant, toutes les mises en œuvre DOIVENT prendre en charge un ensemble minimal d'algorithmes pour assurer l'interopérabilité (cela n'implique pas que la politique locale sur tous les hôtes va nécessairement permettre ces algorithmes). Les algorithmes obligatoires sont spécifiés dans les documents de protocole pertinents.

Des algorithmes, méthodes, formats, et extensions de protocole peuvent être définis dans des documents séparés. Voir plus d'informations à la Section 6, "Désignation des algorithmes".

4.3 Questions de politique

Le protocole permet une pleine négociation des algorithmes et formats de chiffrement, d'intégrité, d'échange de clés, de compression, et de clé publique. Les algorithmes de chiffrement, d'intégrité, de clé publique, et de compression peuvent être différents pour chaque direction.

Les problèmes de politique suivants DEVRAIENT être traités dans les mécanismes de configuration de chaque mise en œuvre :

- Algorithmes de chiffrement, d'intégrité, et de compression séparés pour chaque direction. La politique DOIT spécifier quel est l'algorithme préféré (par exemple, le premier algorithme de la liste dans chaque catégorie).

- Algorithmes de clé publique et méthode d'échange de clés à utiliser pour l'authentification de l'hôte. L'existence de clés d'hôte de confiance pour différents algorithmes de clé publique affecte aussi ce choix.

- Les méthodes d'authentification à exiger par le serveur pour chaque utilisateur. La politique du serveur PEUT exiger plusieurs authentifications pour certains utilisateurs, ou tous. Les algorithmes exigés PEUVENT dépendre de la localisation à partir de laquelle l'utilisateur essaye d'obtenir l’accès.

- Les opérations que l'utilisateur est autorisé à effectuer en utilisant le protocole de connexion. Certains problèmes sont relatifs à la sécurité ; par exemple, la politique NE DEVRAIT PAS permettre au serveur de commencer les sessions ou de traiter les commandes sur la machine du client, et NE DOIT PAS permettre de connexions à l'agent d'authentification tant que la transmission de telles connexions n'a pas été demandée. D'autres problèmes, comme celui des accès TCP/IP qui peuvent être utilisés pour la transmission, et par qui, sont clairement des questions de politique locale. Beaucoup de ces problèmes peuvent impliquer de traverser ou outrepasser des pare-feu, et sont en interrelation avec la politique locale de sécurité.

4.4 Propriétés de sécurité

Le but principal du protocole SSH est d'améliorer la sécurité sur l'Internet. Il tente de le faire d'une façon facile à déployer, même au prix d'une sécurité absolue.

- Tous les algorithmes de chiffrement, d'intégrité, et de clé publique utilisés sont des algorithmes bien connus et bien établis.

- Tous les algorithmes sont utilisés avec des tailles de clés de chiffrement qui sont estimées fournir une protection contre même les plus fortes attaques de cryptanalyse pour des décennies.

- Tous les algorithmes sont négociés, et au cas où certains algorithmes seraient cassés, il est aisé de passer à un autre algorithme sans modifier le protocole de base.

Des concessions spécifiques ont été faites pour rendre plus facile un développement large et rapide. Le cas particulier où cela apparaît est celui de la vérification que la clé d'hôte serveur appartient réellement à l'hôte désiré ; le protocole permet que la vérification ne soit pas faite, mais ceci est NON RECOMMANDÉ. C'est supposé améliorer significativement l'utilisabilité à court terme, jusqu'à ce qu'émergent des infrastructures largement répandues de clé publiques dans l'Internet.

4.5 Prise en charge de la localisation et des jeux de caractères

Pour leur plus grande partie, les protocoles SSH ne passent pas directement du texte qui va être affiché à l'utilisateur. Cependant, il y a quelques endroits où de telles données peuvent être passées. Quand applicable, le jeu de caractères pour les données DOIT être explicitement spécifié. La plupart du temps, le codage ISO-10646 UTF-8 est utilisé [RFC3629]. Lorsque applicable, un champ est aussi fourni pour une étiquette de langage [RFC3066].

Un gros problème est le jeu de caractères de la session interactive. Il n'y a pas de solution claire, car différentes applications peuvent afficher des données dans différents formats. Différents types d'émulation de terminal peuvent aussi être employées chez le client, et le jeu de caractères à utiliser est effectivement déterminé par l'émulation de terminal. Donc, aucun endroit n'est fourni pour spécifier directement le jeu de caractères ou le codage pour les données de session de terminal. Cependant, le type d'émulation de terminal (par exemple, "vt100") est transmis au site distant, et il spécifie implicitement le jeu de caractères et le codage. Les applications utilisent normalement le type de terminal pour déterminer quel jeu de caractères elles utilisent, ou le jeu de caractères est déterminé en utilisant des moyens externes. L'émulation de terminal peut aussi permettre de configurer le jeu de caractères par défaut. Dans tous les cas, le jeu de caractères pour la session de terminal est considéré principalement comme un problème de client.

Les noms internes utilisés pour identifier les algorithmes ou protocoles ne sont normalement jamais affichés aux utilisateurs, et doivent être en US-ASCII.

Les noms d'utilisateur de client et de serveur sont par nature contraints par ce que le serveur est prêt à accepter. Ils peuvent, cependant, être occasionnellement affichés dans des enregistrements, des rapports, etc. Ils DOIVENT être codés en utilisant ISO 10646 UTF-8, mais d'autres codages peuvent être exigés dans certains cas. Il appartient au serveur de décider comment transposer les noms d'utilisateur en noms d'utilisateur acceptés. Une comparaison binaire directe bit par bit est RECOMMANDÉE.

Pour les besoins de localisation, le protocole tente de minimiser le nombre de messages textuels transmis. Lorsque ils sont présents, de tels messages se rapportent normalement aux erreurs, aux informations de débogage, ou à des données configurées en externe. Pour les données qui sont normalement affichées, il DEVRAIT être possible d'aller chercher un message localisé plutôt que le message transmis en utilisant un code numérique. Les messages restants DEVRAIENT être configurables.

5. Représentations des types de données utilisés dans les protocoles SSH

byte : un octet représente une valeur arbitraire de 8 bits. Les données de longueur fixée sont parfois représentées comme une matrice d'octets, écrite byte[n], où n est le nombre d'octets dans la matrice.

boolean : une valeur booléenne est mémorisée comme un seul octet. La valeur 0 représente FAUX, et la valeur 1 représente VRAI. Toutes les valeurs non zéro DOIVENT être interprétées comme VRAI ; cependant, les applications NE DOIVENT PAS mémoriser des valeurs autres que 0 et 1.

uint32 : représente un entier non signé de 32 bits. Mémorisé sur quatre octets en ordre de poids décroissant (ordre des octets du réseau). Par exemple, la valeur 699921578 (0x29b7f4aa) est mémorisée comme 29 b7 f4 aa.

uint64 : représente un entier non signé de 64 bits. Mémorisé sur huit octets en ordre de poids décroissant (ordre des octets du réseau).

string : chaîne binaire de longueur arbitraire. Les chaînes peuvent contenir des données binaires arbitraires, incluant des caractères nuls et des caractères de huit bits. Elles sont mémorisées comme un uint32 contenant sa longueur (nombre d'octets qui suivent) et zéro (= chaîne vide) un ou plusieurs octets qui sont la valeur de la chaîne. On n'utilise pas de terminaison avec des caractères nuls. Les chaînes sont aussi utilisées pour mémoriser du texte. Dans ce cas, l'US-ASCII est utilisé pour les noms internes, et l'ISO-10646 UTF-8 pour le texte qui peut être affiché à l'utilisateur. Le caractère nul de terminaison NE DEVRAIT PAS normalement être mémorisé dans la chaîne. Par exemple: La chaîne US-ASCII "testing" est représentée par 00 00 00 07 t e s t i n g. La transposition UTF-8 n’altère pas le codage des caractères US-ASCII.

mpint : représente des entiers à précision multiple en format de complément à deux, mémorisés comme une chaîne, 8 bits par octet, octet de poids fort en premier. Les nombres négatifs ont la valeur 1 comme bit de poids fort du premier octet de la partition de données. Si le bit de poids fort doit indiquer un nombre positif, le nombre DOIT être précédé par un octet à zéro. Des octets inutiles en tête d'une valeur 0 ou 255 NE DOIVENT PAS être inclus. La valeur zéro DOIT être mémorisée comme une chaîne avec des octets de données à zéro.

Par convention, un nombre qui est utilisé dans des calculs modulaires dans Z_n DEVRAIT être représenté dans la gamme $0 \leq x < n$.

Exemples :

valeur (hex)	**représentation (hex)**
0	00 00 00 00

9a378f9b2e332a7	00 00 00 08 09 a3 78 f9 b2 e3 32 a7
80	00 00 00 02 00 80
-1234	00 00 00 02 ed cc
-deadbeef	00 00 00 05 ff 21 52 41 11

name-list : chaîne contenant une liste de noms séparés par des virgules. Une liste de noms est représentée comme un uint32 contenant sa longueur (nombre d'octets qui suivent) suivi par une liste séparée par des virgules de zéro, un ou plusieurs noms. Un nom DOIT avoir une longueur différente de zéro, et il NE DOIT PAS contenir une virgule (","). Comme c'est une liste de noms, tous les éléments contenus sont des noms et DOIVENT être en US-ASCII. Le contexte peut imposer des restrictions supplémentaires aux noms. Par exemple, les noms dans une name-list peuvent devoir être des identifiants d'algorithme valides (voir la Section 6) ou une liste d'étiquettes de langue de la [RFC3066]. L'ordre des noms dans une liste de noms peut être ou non significatif. Là encore, cela dépend du contexte dans lequel la liste est utilisée. Des caractères nuls NE DOIVENT PAS être utilisés en terminaison, ni pour les noms individuels, ni pour la liste comme un tout.

Exemples :

valeur (hex)	**représentation (hex)**
() (liste de noms vide)	00 00 00 00
("zlib")	00 00 00 04 7a 6c 69 62
("zlib,none")	00 00 00 09 7a 6c 69 62 2c 6e 6f 6e 65

6. Désignation des algorithmes et des méthodes

Les protocoles SSH se réfèrent aux algorithmes ou méthodes particuliers de hachage, chiffrement, intégrité, compression, et échange de clés par noms. Il y a des algorithmes et méthodes standard que toutes les mises en œuvre DOIVENT prendre en charge. Il y a aussi des algorithmes et méthodes qui sont définis dans la spécification du protocole, mais qui sont FACULTATIFS. De plus, on s'attend à ce que certaines organisations veuillent utiliser leurs propres algorithmes ou méthodes.

Dans ce protocole, tous les identifiants d'algorithme et de méthode DOIVENT être en chaînes US-ASCII imprimables, non vides, de pas plus de 64 caractères. Les noms DOIVENT être sensibles à la casse.

Il y a deux formats pour les noms d'algorithmes et de méthodes :

- Les noms qui ne contiennent pas de signe "@" sont réservés pour être alloués par CONSENSUS DE L'IETF. Les exemples incluent "3des-cbc", "sha-1", "hmac-sha1", et "zlib" (les guillemets ne font pas partie du nom). Les noms de ce format ne sont valides que si ils sont préalablement enregistrés par l'IANA. Les noms enregistrés NE DOIVENT PAS contenir de signe "@", virgule (","), espace, caractères de contrôle (codes ASCII 32 ou moins) ou le code ASCII 127 (DEL). Les noms sont sensibles à la casse, et NE DOIVENT PAS faire plus de 64 caractères.

- On peut définir des algorithmes ou méthodes supplémentaires en utilisant des noms de format nom@nom-de-domaine, par exemple, "monchiffre-cbc@exemple.com". Le format de la partie qui précède le signe @ n'est pas spécifié ; cependant, ces noms DOIVENT être des chaînes US-ASCII imprimables, et NE DOIVENT PAS contenir le caractère virgule (","), espace, les caractères de contrôle (codes ASCII 32 ou moins) ou le code ASCII 127 (DEL). Ils DOIVENT avoir seulement un caractère @. La partie qui suit le caractère @ DOIT être un nom de domaine valide, pleinement qualifié [RFC1034] contrôlé par la personne ou organisation qui définit le nom. Les noms sont sensibles à la casse, et NE DOIVENT PAS faire plus de 64 caractères. Il appartient à chaque domaine de gérer son espace de noms local. On devrait noter que ces noms ressemblent aux adresses de messagerie électronique du STD 11 [RFC0822]. C'est une pure coïncidence et n'a rien à voir avec le STD 11 [RFC0822].

7. Numéros de message

Les paquets SSH ont des numéros de message dans la gamme de 1 à 255. Ces numéros ont été alloués comme suit :

Protocole de couche transport :
1 à 19 : couche de transport générique (par exemple, déconnecter, ignorer, débogage, etc.)
20 à 29 : négociation d'algorithme
30 à 49 : spécificités de la méthode d'échange de clé (les numéros peuvent être réutilisés pour des méthodes différentes d'authentification)

Protocole d'authentification d'utilisateur :
50 à 59 : authentification d'utilisateur générique
60 à 79 : spécificités de la méthode d'authentification d'utilisateur (les numéros peuvent être réutilisés pour des méthodes différentes d'authentification)

Protocole de connexion :
80 à 89 : protocole de connexion générique
90 à 127 : messages relatifs au canal

Réservé pour les protocoles de client :
128 à 191 : réservé

Extensions locales :
192 à 255 : extensions locales

8. Considérations relatives à l'IANA

Le présent document fait partie d'un ensemble. Les instructions à l'IANA pour le protocole SSH, tel que définies dans le présent document, dans les [RFC4252], [RFC4253], et [RFC4254], sont détaillées dans la [RFC4250]. Ce qui suit est un bref résumé pour respecter les convenances, mais on notera bien que la [RFC4250] contient les instructions réelles à l'IANA, qui pourront être remplacées à l'avenir.

L'allocation des types de noms suivants dans les protocoles SSH est faite par CONSENSUS DE L'IETF :

- Noms de service
  * Méthodes d'authentification
  * Noms de canaux de protocole de connexion
  * Noms de demande globale de protocole de connexion
  * Noms de demande de canal de protocole de connexion

- Noms de méthode d'échange de clés

- Noms d'algorithme alloué
  * Noms d'algorithme de chiffrement
  * Noms d’algorithme de MAC
  * Noms d'algorithme de clé publique
  * Noms d'algorithme de compression

Ces noms DOIVENT être des chaîne US-ASCII imprimables, et NE DOIVENT PAS contenir les caractères "@", virgule (","), espace, les caractères de contrôle (codes ASCII 32 ou moins) ou le code ASCII 127 (DEL). Les noms sont sensibles à la casse, et NE DOIVENT PAS faire plus de 64 caractères.

Les noms avec le caractère "@" sont des extensions définies en local et ne sont pas contrôlés par l'IANA.

Chaque catégorie de noms figurant ci-dessus a un espace de noms séparé. Cependant, utiliser le même nom dans plusieurs catégories DEVRAIT être évité pour minimiser le risque de confusion.

Les numéros de message (voir la Section 7) dans la gamme de 0 à 191 sont alloués via CONSENSUS DE L'IETF, comme décrit dans la [RFC2434]. Les numéros de message dans la gamme de 192 à 255 (extensions locales) sont réservés pour UTILISATION PRIVÉE, aussi comme décrit dans la [RFC2434].

9. Considérations sur la sécurité

Afin de rendre le corps entier des considérations sur la sécurité plus accessible, les considérations sur la sécurité du transport, de l'authentification, et de la connexion ont été rassemblées ici.

Le protocole de transport [RFC4253] fournit un canal confidentiel sur un réseau non sûr. Il effectue l'authentification de l'hôte serveur, la protection de l'échange de clés, du chiffrement, et de l'intégrité. Il déduit aussi un identifiant de session univoque qui peut être utilisé par les protocoles de niveau supérieur.

Le protocole d'authentification [RFC4252] fournit une suite de mécanismes qui peuvent être utilisés pour authentifier l'utilisateur client auprès du serveur. Les mécanismes individuels spécifiés dans le protocole d'authentification utilisent l'identifiant de session fourni par le protocole de transport et/ou dépendent des garanties de sécurité et d'intégrité du protocole de transport.

Le protocole de connexion [RFC4254] spécifie un mécanisme pour multiplexer plusieurs flux (canaux) de données sur le transport confidentiel et authentifié. Il spécifie aussi les canaux pour accéder à une coquille interactive, pour transmettre par l'intermédiaire de mandataire divers protocoles externes sur le transport sûr (incluant des protocoles TCP/IP arbitraires) et pour accéder à des sous systèmes sécurisés sur l'hôte serveur.

9.1 Génération de numéros pseudo aléatoires

Ce protocole lie chaque clé de session à la session en incluant des données aléatoires spécifiques de la session dans le hachage utilisé pour produire les clés de session. Un soin particulier devrait être apporté à s'assurer que tous les nombres aléatoires sont de bonne qualité. Si les données aléatoires (par exemple, les paramètres Diffie-Hellman (DH)) sont pseudo aléatoires, le générateur de nombres pseudo aléatoire devrait être cryptographiquement sûr (c'est-à-dire que son prochain résultat ne puisse pas être deviné même connaissant tous les résultats précédents) et de plus, une entropie appropriée doit être ajoutée au générateur de nombres pseudo aléatoires. La [RFC4086] propose des suggestions pour les sources de nombres aléatoires et d'entropie. Les mises en œuvre devraient noter l'importance de l'entropie et l'avertissement anecdotique très significatif sur la difficulté de mettre proprement en œuvre les fonctions de génération de nombres pseudo aléatoires.

La quantité d'entropie disponible à un certain client ou serveur peut parfois être moindre que ce qui est requis. Dans ce cas, on doit soit avoir recours à la génération de nombre pseudo aléatoire sans considération de l'entropie insuffisante, soit refuser le protocole. Cette dernière solution est préférable.

9.2 Filtrage des caractères de contrôle

Lors de l'affichage d'un texte à un utilisateur, comme des messages d'erreur ou de débogage, le logiciel client DEVRAIT remplacer tous les caractères de contrôle (sauf les tabulations, les retours chariot, et les sauts à la ligne) par des séquences sûres pour éviter des attaques par envoi de caractères de contrôle terminaux.

9.3 Transport

9.3.1 Confidentialité

Il sort du domaine d'application du présent document et du mandat du groupe de travail Secure Shell d'analyser ou recommander des chiffrements spécifiques autres que ceux qui ont été établis et acceptés dans l'industrie. Au moment de la rédaction de ce document, les chiffrements couramment utilisés incluent 3DES, ARCFOUR, twofish, serpent, et blowfish. AES a été publié par l'Institut américain des normes et technologies comme normes fédérales US de traitement de l'information [FIPS-197], et la communauté cryptographique a accepté AES aussi. Comme toujours, les mises en œuvre et les utilisateurs devraient vérifier dans la littérature courante qu'aucune vulnérabilité récente n'a été découverte dans les chiffrements au sein de ces produits. Les mises en œuvre devraient aussi vérifier quels chiffrements sont considérés comme étant relativement plus forts que d'autres et devraient recommander leur utilisation aux utilisateurs plutôt que celle des chiffrements relativement plus faibles. Il serait considéré de bonne forme qu'une mise en œuvre notifie poliment et de façon non obstructive à un utilisateur qu'un chiffrement plus fort est disponible et devrait être utilisé quand un chiffrement faible est activement choisi.

Le chiffrement "none" est fourni pour le débogage et, excepté pour cet objet, NE DEVRAIT PAS être utilisé. Ses propriétés cryptographiques sont suffisamment décrites dans la [RFC2410], qui montre que son utilisation ne correspond pas aux intentions du présent protocole.

Les mérites relatifs de ces chiffrements et des autres peut aussi se trouver dans la littérature courante. Deux références qui peuvent fournir des informations sur le sujet sont [SCHNEIER] et [KAUFMAN]. Toutes deux décrivent le mode CBC de fonctionnement de certains chiffrements et les faiblesses de ce schéma. Pour l'essentiel, ce mode est théoriquement vulnérable aux attaques de texte chiffré choisi à cause de la forte prédictibilité du début de la séquence de paquets.

Cependant, cette attaque est réputée difficile et n'est pas considérée comme pleinement praticable, en particulier si des tailles de bloc relativement longues sont utilisées.

De plus, une autre attaque de mode CBC peut être atténuée par l'insertion de paquets contenant SSH_MSG_IGNORE. Sans cette technique, une attaque spécifique peut réussir. Pour que cette attaque (connue généralement sous le nom d'attaque Rogaway [ROGAWAY], [DAI], [BELLARE]) fonctionne, l'attaquant va avoir besoin de connaître la valeur d'initialisation (IV) du prochain bloc qui va être chiffré. En mode CBC, c'est le résultat du chiffrement du bloc précédent. Si l'attaquant n'a aucun moyen de voir le paquet (c'est-à-dire, si il est dans les mémoires tampon internes de la mise en œuvre SSH ou même dans le noyau) cette attaque ne va pas fonctionner. Si le dernier paquet a été envoyé sur le réseau (c'est-à-dire que l'attaquant y a accès) il peut alors utiliser l'attaque.

Dans le meilleur cas, une mise en œuvre n'aura besoin d'ajouter un paquet supplémentaire que si le paquet a été envoyé sur le réseau et si il n'y a pas d'autre paquet en attente de transmission. Les mises en œuvre peuvent souhaiter vérifier si il y a des paquets non envoyés qui attentent leur transmission ; malheureusement, il n'est normalement pas facile d'obtenir ces informations du noyau ou des mémoires tampon. Si il n'y a pas de paquet non envoyé, un paquet contenant SSH_MSG_IGNORE DEVRAIT être envoyé. Si un nouveau paquet est ajouté au flux chaque fois que l'attaquant connaît l'IV qui est supposée être utilisée pour le prochain paquet, l'attaquant ne sera alors pas capable de deviner l'IV correcte, et l'attaque échouera donc toujours.

Par exemple, considérons le cas suivant :

Client **Serveur**
TCP(seq=x, len=500) contient enregistrement 1 ---->
[500 ms passent, pas de ACK]
TCP(seq=x, len=1000) contient enregistrements 1, 2 ---->
ACK

1. L'algorithme de Nagle + les retransmissions TCP signifient que les deux enregistrements sont confondus en un seul segment TCP.
2. L' enregistrement 2 n'est pas au début du segment TCP et ne le sera jamais parce qu'il est acquitté.
3. Cependant, l'attaque est possible parce que l' enregistrement 1 a déjà été vu.

Comme l'indique cet exemple, il n'est pas sûr d'utiliser l'existence de données non purgées dans les mémoires tampon TCP comme indication de si un paquet vide est nécessaire car quand le second write() est effectué, les mémoires tampon vont contenir l'enregistrement 1 non acquitté.

Par ailleurs, il est parfaitement sûr d'avoir la situation suivante :

Client **Serveur**
TCP(seq=x, len=500) contient SSH_MSG_IGNORE ---->
TCP(seq=y, len=500) contient des données ---->

Pourvu que la IV pour le second enregistrement SSH soit fixée après que les données pour le paquet de données sont déterminées, ce qui suit devrait être effectué :
lecture par l'utilisateur
paquet de chiffrement nul
chiffrement du paquet de données

9.3.2 Intégrité des données

Ce protocole ne permet pas que le mécanisme d'intégrité des données soit désactivé. Les mises en œuvre DEVRAIENT être conscientes que l'exposition de cette caractéristique n'est faite que à des fins de débogage. Les utilisateurs et administrateurs DEVRAIENT être avertis explicitement chaque fois que le MAC "none" est activé.

Tant que le MAC "none" n'est pas utilisé, ce protocole assure l'intégrité des données.

Comme les MAC utilisent un numéro de séquence de 32 bits, ils peuvent commencer à laisser échapper des informations après l'envoi de 2**32 paquets. Cependant, suivant les recommandations, les changements de clés devraient empêcher cette attaque. Le protocole de transport [RFC4253] recommande de changer les clés après un giga octet de données, et le plus petit paquet possible est de 16 octets. Donc, le changement de clés DEVRAIT se produire au plus après 2**28 paquets.

9.3.3 Répétition

L'utilisation d'un MAC autre que "none" assure la protection de l'intégrité et l'authentification. De plus, le protocole de transport fournit un identifiant unique de session (lié en partie aux données pseudo aléatoires qui font partie de l'algorithme et du processus d'échange de clés) qui peut être utilisé par des protocoles de niveau supérieur pour lier les données à une certaine session et empêcher la répétition des données provenant de sessions antérieures. Par exemple, le protocole d'authentification ([RFC4252]) utilise cela pour empêcher la répétition des signatures provenant de sessions antérieures. Parce que les échanges d'authentification de clé publique sont liés cryptographiquement à la session (c'est-à-dire, à l'échange de clés initial) ils ne peuvent pas être répétés avec succès dans d'autres sessions. Noter que l'identifiant de session peut être rendu public sans dommage pour la sécurité du protocole.

Si deux sessions ont le même identifiant de session (hachage des échanges de clé) alors les paquets de l'une peuvent être répétés dans l'autre. On doit souligner que les chances d'une telle occurrence sont, inutile de le dire, minimales quand on utilise les méthodes modernes de cryptographie. C'est particulièrement vrai quand on spécifie de plus grands résultats de fonction de hachage et des paramètres DH.

La détection de répétitions en utilisant des numéros de séquence à accroissement monotone comme entrée dans le MAC, ou HMAC dans certains cas, est décrite dans les [RFC2085], [RFC2246], [RFC2743], [RFC1964], [RFC2025], et [RFC4120]. La construction sous-jacente est discutée dans la [RFC2104]. Pour l'essentiel, un numéro de séquence différent dans chaque paquet assure qu'au moins cette entrée à la fonction de MAC sera unique et va fournir un résultat de MAC non récurrent qui n'est pas prévisible par un attaquant. Si la session reste active assez longtemps, cependant, ce numéro de séquence va revenir à zéro. Cet événement peut fournir à un attaquant une opportunité de répéter un paquet enregistré précédemment avec un numéro de séquence identique mais seulement si les homologues n'ont pas changé de clés depuis la transmission du premier paquet avec ce numéro de séquence. Si les homologues ont changé de clés, la répétition sera détectée car la vérification du MAC va échouer. Pour cette raison, on doit souligner que les homologues DOIVENT changer de clés avant que se produise un retour à zéro des numéros de séquence. Naturellement, si un attaquant tente de répéter un paquet capturé avant que les homologues aient changé de clés, le receveur du paquet dupliqué ne sera pas capable de valider le MAC et il sera éliminé. La raison pour laquelle le MAC va échouer est que le receveur va formuler un MAC sur la base du contenu du paquet, du secret partagé, et du numéro de séquence attendu. Comme le paquet répété ne va pas utiliser le numéro de séquence attendu (le numéro de séquence du paquet répété aura déjà été passé par le receveur) le MAC calculé ne va pas correspondre au MAC reçu avec le paquet.

9.3.4 Interposition

Le présent protocole ne fait aucune hypothèse ni ne prend aucune disposition pour une infrastructure ou moyens pour la distribution des clés publiques des hôtes. Il est prévu que ce protocole soit parfois utilisé sans vérifier d'abord l'association entre la clé d'hôte serveur et le nom d'hôte serveur. Un tel usage est vulnérable aux attaques par interposition. Ce paragraphe décrit cela et encourage les administrateurs et utilisateurs à comprendre l'importance de la vérification de cette association avant l'initialisation de toute session.

Il y a trois cas d'attaques par interposition à considérer. Le premier est lorsque un attaquant place un appareil entre le client et le serveur avant que la session soit initiée. Dans ce cas, l'appareil d'attaque essaye d'imiter le serveur légitime et va offrir sa clé publique au client quand il initie une session. Si il devait offrir la clé publique au serveur, il ne serait alors pas capable de déchiffrer ou signer les transmissions entre le serveur légitime et le client sauf si il a aussi accès à la clé privée de l'hôte. L'appareil d'attaque va aussi, simultanément à cela, initier une session au serveur légitime, se faisant passer pour le client. Si la clé publique du serveur a été distribuée de façon sûre au client avant cette initialisation de session, la clé offerte au client par l'appareil d'attaque ne va pas correspondre à la clé mémorisée chez le client. Dans ce cas, l'utilisateur DEVRAIT recevoir un avertissement disant que la clé d'hôte offerte ne correspond pas à la clé d'hôte mise en antémémoire chez le client. Comme décrit au paragraphe 4.1, l'utilisateur peut être libre d'accepter la nouvelle clé et de continuer la session. Il est RECOMMANDÉ que l'avertissement fournisse des informations suffisantes à l'utilisateur de l'appareil client afin qu'il puisse prendre une décision informée. Si l'utilisateur choisit de continuer la session avec la clé publique mémorisée du serveur (pas la clé publique offerte au début de la session) les données spécifiques de session entre l'attaquant et le serveur seront différentes entre la session de client à attaquant et les sessions d'attaquant à serveur du fait de l'aléa discuté plus haut. À partir de cela, l'attaquant ne sera pas capable de faire fonctionner son attaque parce qu'il ne sera pas capable de signer correctement les paquets contenant ces données spécifiques de session à partir du serveur, car il n'a pas la clé privée de ce serveur.

Le second cas devrait être considéré comme similaire au premier en ce qu'il arrive aussi au moment de la connexion, mais ce cas souligne le besoin d'une distribution sûre des clés publiques de serveur. Si les clés publiques du serveur ne sont pas distribuées de façon sûre, le client ne peut alors pas savoir si il parle au serveur prévu. Un attaquant peut utiliser les techniques d'ingénierie sociale pour passer des clés de serveur à des utilisateurs non soupçonneux et peut alors placer un appareil d'attaque par interposition entre le serveur légitime et les clients. Si on permet que cela arrive, les clients vont former des sessions de client à attaquant, et l'attaquant va former des session d'attaquant à serveur et sera capable de

surveiller et manipuler tout le trafic entre les clients et les serveurs légitimes. Les administrateurs de serveurs sont encouragés à rendre des empreintes digitales de clé d'hôte disponibles pour vérifier par des moyens dont la sécurité ne repose pas sur l'intégrité des clés d'hôte réelles. Des mécanismes possibles sont discutés au paragraphe 4.1 et peuvent aussi inclure des pages de la Toile sécurisées, des feuilles de papier physiques, etc. Les mises en œuvre DEVRAIENT fournir des recommandations sur la meilleure façon de faire cela avec leur mise en œuvre. Comme le protocole est extensible, de futures extensions au protocole pourront fournir de meilleurs mécanismes pour traiter le besoin de connaître la clé d'hôte de serveur avant de se connecter. Par exemple, rendre l'empreinte digitale de la clé d'hôte disponible à travers une recherche sur le DNS, ou en utilisant Kerberos ([RFC4120]) sur GSS-API ([RFC1964]) durant l'échange de clés pour authentifier le serveur sont des possibilités.

Dans le troisième cas d'attaque par interposition, les attaquants peuvent tenter de manipuler les paquets en transit entre les homologues après l'établissement de la session. Comme décrit au paragraphe 9.3.3, une attaque réussie de cette nature est très improbable. Comme au paragraphe 9.3.3, ce raisonnement fait l'hypothèse que le MAC est sûr et qu'il est infaisable de construire des entrées à un algorithme de MAC pour donner un résultat connu. Ceci est discuté plus en détails à la Section 6 de la [RFC2104]. Si l'algorithme de MAC a une vulnérabilité ou est assez faible, l'attaquant peut alors être capable de spécifier certaines entrées pour donner un MAC connu. Avec cela, il peut être capable d'altérer le contenu d'un paquet en transit. Autrement, l'attaquant peut être capable d'exploiter la vulnérabilité ou faiblesse de l'algorithme pour trouver le secret partagé en étudiant les MAC provenant des paquets capturés. Dans l'un et l'autre de ces cas, un attaquant pourrait construire un ou des paquets qui pourraient être insérés dans un flux SSH. Pour empêcher cela, les mises en œuvre sont invitées à utiliser les algorithmes de MAC couramment acceptés, et les administrateurs sont encouragés à surveiller la littérature courante et les discussions de cryptographie pour s’assurer qu'ils n'utilisent pas un algorithme de MAC auquel on aurait récemment trouvé des vulnérabilités ou faiblesses.

En résumé, l'utilisation de ce protocole sans une association fiable de lien entre un hôte et ses clés d'hôte est par nature non sûr et N'EST PAS RECOMMANDÉ. Cependant, il peut être nécessaire dans des environnement non critiques de sécurité, et il va quand même fournir une protection contre les attaques passives. Les mises en œuvre de protocoles et applications fonctionnant par dessus ce protocole devraient garder cette possibilité en mémoire.

9.3.5 Déni de service

Ce protocole est conçu pour être utilisé sur un transport fiable. Si des erreurs de transmission ou des manipulations de message se produisent, la connexion est fermée. La connexion DEVRAIT être rétablie si cela se produit. Des attaques de déni de service de ce type (coupe réseau) sont presque impossibles à éviter.

De plus, ce protocole est vulnérable aux attaques de déni de service parce qu'un attaquant peut forcer le serveur à effectuer des tâches intensives de CPU et de mémoire pour l'établissement de connexion et l'échange de clés sans s'authentifier. Les mises en œuvre DEVRAIENT fournir des caractéristiques qui rendent cela plus difficile, par exemple, de ne permettre de connexions que d'un sous ensemble de clients connus pour avoir des utilisateurs valides.

9.3.6 Canaux couverts

Le protocole n'a pas été conçu pour éliminer les canaux couverts. Par exemple, le bourrage, les messages SSH_MSG_IGNORE, et plusieurs endroits du protocole peuvent être utilisés pour passer des informations couvertes, et le receveur n'a pas de moyen fiable pour vérifier si de telles informations sont envoyées.

9.3.7 Secret vers l'avant

On devrait noter que les échanges de clé Diffie-Hellman peuvent fournir le secret parfait vers l'avant (PFS, *perfect forward secrecy*). PFS est essentiellement défini comme la propriété cryptographique d'un protocole d'établissement de clé dans lequel la compromission d'une clé de session ou d'une clé privée à long terme après une certaine session ne cause pas la compromission d'une session antérieure [ANSI-T1.523-2001]. Les sessions SSH résultant d'un échange de clés qui utilise les méthodes Diffie-Hellman décrites à la Section 8 "Échange de clé Diffie-Hellman" de la [RFC4253] (incluant "diffie-hellman-group1-sha1" et "diffie-hellman-group14-sha1") sont sûres même si le matériel de chiffrement privé/authentification est ensuite révélé, mais pas si les clés de session sont révélées. Donc, selon cette définition de PFS, SSH a bien le PFS. Cependant, cette propriété n'est pas commutée aux applications ou protocoles qui utilisent SSH comme transport. La couche de transport de SSH fournit la confidentialité pour l'authentification de mot de passe et autres méthodes qui s'appuient sur des données secrètes.

Bien sûr, si les paramètres DH privés pour le client et le serveur sont révélés, la clé de session est révélée, mais ces éléments peuvent être éliminés après l'achèvement de l'échange de clés. On soulignera qu'il ne devrait pas être permis que ces éléments se terminent sur un espace d'échange et qu'ils devraient être éliminés de la mémoire aussitôt que s'achève l'échange de clés.

9.3.8 Ordre des méthodes d'échange de clés

Comme il est mentionné au paragraphe 7.1 "Négociation d'algorithme" de la [RFC4253], chaque appareil va envoyer une liste de méthodes préférées pour l'échange de clés. La méthode préférée est la première de la liste. Il est RECOMMANDÉ que les algorithmes soient triés par force de chiffrement, le plus fort en premier. Des lignes directrices supplémentaires sur ce point sont données dans la [RFC3766].

9.3.9 Analyse de trafic

La surveillance passive de tout protocole peut donner à un attaquant des informations sur la session, l'utilisateur, ou des informations spécifiques du protocole qu'il ne serait par ailleurs pas capable de recueillir. Par exemple, on a montré que l'analyse de trafic d'une session SSH peut donner des informations sur la longueur du mot de passe - [Openwall] et [USENIX]. Les mises en œuvre devraient utiliser le paquet SSH_MSG_IGNORE, avec l'inclusion de bourrages de longueur aléatoire, pour déjouer les tentatives d'analyse de trafic. D'autres méthodes peuvent aussi être trouvées et mises en œuvre.

9.4 Protocole d'authentification

L'objet de ce protocole est d'effectuer l'authentification de l'utilisateur du client. On suppose que cela fonctionne sur un protocole de couche de transport sûr, qui a déjà authentifié la machine du serveur, établi un canal de communications chiffré, et calculé un identifiant de session unique pour cette session.

Plusieurs méthodes d'authentification avec des caractéristique de sécurité différentes sont permises. Il appartient à la politique locale du serveur de décider quelles méthodes (ou combinaisons de méthodes) elle veut accepter pour chaque utilisateur. L'authentification n'est pas plus forte que la plus faible combinaison permise.

Le serveur peut entrer dans une période de sommeil après des tentatives répétées d'authentification qui ont échoué pour rendre la recherche de clés plus difficile aux attaquants. On devrait veiller à ce que ceci ne devienne pas un vecteur d'auto déni de service.

9.4.1 Transport faible

Si la couche de transport ne fournit pas la confidentialité, les méthodes d'authentification qui s'appuient sur des données secrètes DEVRAIENT être désactivées. Si elle ne fournit pas une protection forte de l'intégrité, les demandes de changement des données d'authentification (par exemple, un changement de mot de passe) DEVRAIENT être désactivées pour empêcher un attaquant de modifier le texte chiffré sans être remarqué, ou de rendre les nouvelles données d'authentification inutilisables (déni de service).

L'hypothèse ci-dessus, que le protocole d'authentification ne fonctionne que sur un transport sûr qui a préalablement authentifié le serveur, est très importante. Les gens qui déploient SSH doivent se rappeler les conséquences des attaques par interposition si le client n'a pas une très forte association a priori du serveur et de la clé d'hôte de ce serveur. Précisément, pour le cas du protocole d'authentification, le client peut former une session avec un appareil d'attaque par interposition et divulguer les accréditifs d'utilisateur comme leur nom d'utilisateur et leur mot de passe. Même dans les cas d'authentification où aucun accréditif d'utilisateur n'est divulgué, un attaquant peut quand même obtenir des informations qu'il ne devrait pas avoir en capturant les touches frappées un peu de la même façon que fonctionne un pot de miel.

9.4.2 Messages de débogage

Une attention particulière devrait être apportée à la conception des messages de débogage. Ces messages peuvent révéler des quantités surprenantes d'informations sur l'hôte si ils ne sont pas conçus de façon appropriée. Les messages de débogage peuvent être désactivés (durant la phase d'authentification de l'utilisateur) si une forte sécurité est requise. Les administrateurs de machines hôtes devraient tout tenter pour compartimenter tous les messages de notification d'événements et les protéger contre l'observation non voulue. Les développeurs devraient être conscients de la nature sensible de certains des messages d'événement normal et de débogage, et peuvent vouloir fournir des directives aux administrateurs sur la façon de garder ces informations à l'abri des personnes non autorisées. Les développeurs devraient envisager de minimiser la quantité d'informations sensibles que peuvent obtenir les utilisateurs durant la phase d'authentification, en accord avec les politiques locales. Pour cette raison, il est RECOMMANDÉ que les messages de débogage soient initialement désactivés au moment du déploiement et qu'il faille une décision active de l'administrateur pour permettre qu'ils soient activés. Il est aussi RECOMMANDÉ qu'un message exprimant ce souci soit présenté à l'administrateur d'un système quand est entreprise l'action d'activation des messages de débogage.

9.4.3 Politique de sécurité locale

Les mises en œuvre DOIVENT s'assurer que les accréditifs fournis valident l'utilisateur prétendu et aussi DOIVENT assurer que la politique locale du serveur permet à l'utilisateur l'accès demandé. En particulier, à cause de la nature flexible du protocole de connexion SSH, il peut n'être pas possible de déterminer la politique de sécurité locale, si il en est une, qui devrait s'appliquer au moment de l'authentification parce que la sorte de service demandé n'est pas claire à cet instant. Par exemple, la politique locale peut permettre à un usager d'accéder à des fichiers sur le serveur, mais pas de commencer une coquille interactive. Cependant, durant le protocole d'authentification, on ne sait pas si l'utilisateur va accéder aux fichiers, tenter d'utiliser une coquille interactive, ou même les deux. En tous cas, lorsque une politique de sécurité locale existe pour l'hôte serveur, elle DOIT être appliquée correctement.

Les mises en œuvre sont encouragées à fournir une politique locale par défaut et à faire connaître ses paramètres aux administrateurs et utilisateurs. À la discrétion des mises en œuvre, cette politique par défaut peut être sur une ligne de tout passe où il n'y a pas de restriction placée sur les utilisateurs, ou elle peut être sur une ligne excessivement restrictive, auquel cas, les administrateurs vont devoir faire des changements actifs aux paramètre initiaux par défaut pour satisfaire leurs besoins. Autrement, ce peut être une tentative pour fournir quelque chose de pratique et immédiatement utile aux administrateurs du système afin qu'ils n'aient pas grand chose à faire pour que SSH fonctionne. Quel que soit le choix effectué, il doit être appliqué comme exigé ci-dessus.

9.4.4 Authentification de clé publique

L'utilisation de l'authentification de clé publique suppose que l'hôte client n'a pas été compromis. Elle suppose aussi que la clé privée de l'hôte serveur n'a pas été compromise.

Ce risque peut être atténué par l'utilisation de mots de passe sur les clés privées ; cependant, ce n'est pas une politique qu'on peut mettre en application. L'utilisation de cartes à mémoire, ou autre technologie pour rendre applicables une politique est suggérée.

Le serveur pourrait exiger l'authentification par mot de passe et par clé publique ; cependant, ceci exige que le client expose son mot de passe au serveur (voir au paragraphe suivant.)

9.4.5 Authentification de mot de passe

Le mécanisme de mot de passe, comme spécifié dans le protocole d'authentification, suppose que le serveur n'a pas été compromis. Si le serveur a été compromis, utiliser l'authentification de mot de passe va révéler une combinaison valide de nom d'utilisateur/mot de passe à l'attaquant, qui peut conduire à d'autres compromissions.

Cette vulnérabilité peut être atténuée en utilisant une autre forme d'authentification. Par exemple, l'authentification de clé publique ne fait pas d'hypothèse sur la sécurité chez le serveur.

9.4.6 Authentification fondée sur l'hôte

L'authentification fondée sur l'hôte suppose que le client n'a pas été compromis. Il n'y a pas de stratégie d'atténuation, autre que d'utiliser l'authentification fondée sur l'hôte en combinaison avec une autre méthode d'authentification.

9.5 Protocole de connexion

9.5.1 Sécurité de point d'extrémité

La sécurité du point d'extrémité est supposée par le protocole de connexion. Si le serveur a été compromis, toute session terminale, transmission par un accès ou tous systèmes accédés sur l'hôte sont compromis. Il n'y a pas de facteurs d'atténuation pour cela.

Si le client a été compromis, et si le serveur échoue à arrêter l'attaquant au protocole d'authentification, tous les services exposés (comme sous systèmes ou par la transmission) seront vulnérables à l'attaque. Les mises en œuvre DEVRAIENT fournir des mécanismes pour que les administrateurs contrôlent quels services sont exposés pour limiter la vulnérabilité des autres services. Ces contrôles peuvent inclure de contrôler quelles machines et quels accès peuvent être ciblés dans les opérations d'accès/transmission, quels utilisateurs sont autorisés à utiliser des facilités de coquille interactive, ou quels utilisateurs sont autorisés à utiliser les sous systèmes exposés.

9.5.2 Transmission par mandataire

Le protocole de connexion SSH permet la transmission par mandataire des autres protocoles comme SMTP, POP3, et HTTP. Ceci peut être un problème pour les administrateurs de réseaux qui souhaitent contrôler l'accès de certaines applications par des utilisateurs situés en dehors de leur localisation physique. Essentiellement, la transmission de ces protocoles peut violer les politiques de sécurité spécifiques du site, car elle peuvent être tunnelées de façon non détectée à travers un pare-feu. Les mises en œuvre DEVRAIENT fournir un mécanisme administratif pour contrôler la fonction de transmission par mandataire afin que les politiques de sécurité spécifiques du site puissent être observées.

De plus, une fonction de transmission inverse par mandataire est disponible, qui là encore, peut être utilisée pour outrepasser les contrôles de pare-feu.

Comme indiqué ci-dessus, la sécurité de point d'extrémité est supposée durant les opérations de transmission par mandataire. Les défaillances de la sécurité de point d'extrémité vont compromettre toutes les données passées sur la transmission par mandataire.

9.5.3 Transmission X11

Une autre forme de transmission par mandataire fournie par le protocole de connexion SSH est la transmission du protocole X11. Si la sécurité de point d'extrémité a été compromise, la transmission X11 peut permettre des attaques contre le serveur X11. Les utilisateurs et administrateurs devraient évidemment utiliser des mécanismes appropriés de sécurité X11 pour empêcher l'utilisation non autorisée du serveur X11. Les mises en œuvre, administrateurs, et utilisateurs qui souhaitent explorer plus avant les mécanismes de sécurité de X11 sont invités à lire [SCHEIFLER] et à analyser les problèmes rapportés précédemment avec les interactions entre la transmission SSH et X11 dans les vulnérabilités VU#363181 et VU#118892 du [CERT].

La transmission de l'affichage X11 avec SSH, n'est par elle-même pas suffisante pour corriger les problèmes bien connus de la sécurité de X11 [VENEMA]. Cependant, la transmission de l'affichage X11 dans SSH (ou d'autres protocoles sûrs) combinée avec des affichages réels et des pseudo affichages qui n'acceptent les connexions que sur les mécanismes IPC locaux autorisés par des listes de permissions ou de contrôle d'accès (ACL, *access control list*) corrige beaucoup des problèmes de sécurité d'X11, pour autant que le MAC "none" n'est pas utilisé. Il est RECOMMANDÉ que les mises en œuvre d'affichage X11 permettent par défaut l'ouverture de l'affichage X11 seulement sur l'IPC local. Il est RECOMMANDÉ que les mises en œuvre de serveur SSH qui prennent en charge la transmission X11 permettent par défaut l'ouverture de l'affichage seulement sur l'IPC local. Sur les systèmes d'un seul utilisateur, il peut être raisonnable de permette par défaut l'ouverture de l'affichage local sur TCP/IP.

Les mises en œuvre du protocole de transmission X11 DEVRAIENT appliquer le mécanisme de cookie magique de vérification de droits d'accès, comme décrit dans la [RFC4254], comme mécanisme supplémentaire pour empêcher l'utilisation non autorisée du mandataire.

10. Références

10.1 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)

[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)

[RFC4254] T. Ylonen et C. Lonvick, éd., "Protocole de connexion Secure Shell (SSH)", janvier 2006. *(P.S. ; MàJ par* RFC8308)

10.2 Références pour information

[T1.523-2001] American National Standards Institute, Inc., "Telecom Glossary 2000", ANSI T1.523-2001, février 2001.

[BELLARE] Bellaire, M., Kohno, T., et C. Namprempre, "Authenticated Encryption in SSH: Fixing the SSH Binary Packet Protocol", Proceedings of the 9th ACM Conference on Computer et Communications Security, septembre 2002.

[CERT] CERT Coordination Center, The., "http://www.cert.org/nav/index_red.html ".

[DAI] Dai, W., "An attack contre SSH2 protocol", message au groupe de travail SECSH ietf-ssh@netbsd.org ftp://ftp.ietf.org/ietf-mail-archive/secsh/2002-02.mail, février 2002.

[FIPS-180-2] US National Institute of Standards et Technology, "Secure Hash Standard (SHS)", Federal Information Processing Standards Publication 180-2, août 2002.

[FIPS-186-2] US National Institute of Standards et Technology, "Digital Signature Standard (DSS)", Federal Information Processing Standards Publication 186-2, janvier 2000.

[FIPS-197] US National Institute of Standards et Technology, "Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, novembre 2001.

[KAUFMAN] Kaufman, C., Perlman, R., et M. Speciner, "Network Security: PRIVATE Communication in a PUBLIC World", Prentice Hall Publisher, 1995.

[Openwall] Solar Designer et D. Song, "SSH Traffic Analysis Attacks", Presentation given at HAL2001 et NordU2002 Conferences, Sept 2001.

[RFC0822] D. Crocker, "Norme pour le format des messages de texte de l'ARPA-Internet", STD 11, août 1982. (*Obsolète, voir RFC*5322)

[RFC0854] J. Postel et J. Reynolds," Spécification du protocole TELNET", STD 8, mai 1983.

[RFC1034] P. Mockapetris, "Noms de domaines - Concepts et facilités", STD 13, novembre 1987. (*MàJ par* RFC1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, 8020, 8482)

[RFC1282] B. Kantor, "La commande Rlogin de BSD", décembre 1991. *(Information)*

[RFC1964] J. Linn, "Mécanisme GSS-API de Kerberos version 5", juin 1996. (*MàJ par* RFC4121 et RFC6649)

[RFC2025] C. Adams, "Mécanisme simple de GSS-API à clé publique (SPKM)", octobre 1996. *(P.S.)*

[RFC2085] M. Oehler, R. Glenn, "Authentification IP par HMAC-MD5 avec prévention de répétition", février 1997. *(P.S.)*

[RFC2104] H. Krawczyk, M. Bellare et R. Canetti, "HMAC : Hachage de clés pour l'authentification de message", février 1997.

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

[RFC2410] R. Glenn, S. Kent, "L'algorithme de chiffrement NULL et son utilisation avec IPsec", novembre 1998. *(P.S.)*

[RFC2743] J. Linn, "Interface générique de programme d'application de service de sécurité, version 2, mise à jour 1", janvier 2000. (*MàJ par* RFC5554)

[RFC3766] H. Orman, P. Hoffman, "Détermination de la force des clés publiques utilisées pour l'échange de clés symétriques", avril 2004. (BCP0086)

[RFC4086] D. Eastlake 3rd, J. Schiller, S. Crocker, "Exigences d'aléa pour la sécurité", juin 2005. (*Remplace* RFC1750) (BCP0106)

[RFC4120] C. Neuman et autres, "Service Kerberos d'authentification de réseau (V5)", juillet 2005. (*MàJ par* RFC4537, 5021, 6649, 7751, 8062, 8129, 8429)

[ROGAWAY] Rogaway, P., "Problems with Proposed IP Cryptography", Unpublished paper http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt , 1996.

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

[SCHNEIER] Schneier, B., "Applied Cryptography Second Edition: protocols algorithms et source in code in C", John Wiley et Sons, New York, NY, 1996.

[USENIX] Song, X.D., Wagner, D., et X. Tian, "Timing Analysis of Keystrokes et SSH Timing Attacks", Paper given at 10th USENIX Security Symposium, 2001.

[VENEMA] Venema, W., "Murphy's Law et Computer Security", Proceedings of 6th USENIX Security Symposium, San Jose CA http://www.usenix.org/publications/library/proceedings/sec96/venema.html , juillet 1996.

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

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.