Groupe de travail Réseau
Request for Comments : 4895
Catégorie : Sur la voie de la normalisation

Traduction Claude Brière de L'Isle

M. Tuexen, Muenster Univ. of Applied Sciences
R. Stewart, Cisco Systems, Inc.
P. Lei, Cisco Systems, Inc.
E. Rescorla, RTFM, Inc.

août 2007

Tronçons authentifiés pour le protocole de transmission de contrôle de flux (SCTP)

Résumé
Le présent document décrit un nouveau type de tronçon, plusieurs paramètres, et des procédures pour le protocole de transmission de contrôle de flux (SCTP, *Stream Control Transmission Protocol*). Ce nouveau type de tronçon peut être utilisé pour authentifier les tronçons SCTP en utilisant des clés partagées entre l'envoyeur et le receveur. Les nouveaux paramètres sont utilisés pour établir les clés partagées.

Table des matières

1. Introduction

SCTP utilise des étiquettes de vérification de 32 bits pour se protéger contre les attaques aveugles. Ces valeurs ne sont pas changées durant la vie d'une association SCTP.

En recherchant de nouvelles extensions SCTP, il est nécessaire d'avoir une méthode pour prouver qu'un ou des tronçons SCTP ont réellement été envoyés par l'homologue original qui a commencé l'association et non par un attaquant malveillant.

L'utilisation de la sécurité de la couche transport (TLS, *Transport Layer Security*) comme défini dans la [RFC3436], n'aide pas parce que cela sécurise seulement les données d'utilisateur SCTP.

Donc, on présente une extension SCTP qui fournit un mécanisme pour déduire des clés partagées pour chaque association. Ces clés partagées d'association sont déduites des clés partagées de paire de points d'extrémité qui sont configurées et pourraient être vides, et des données échangées durant l'établissement de l'association SCTP.

L'extension présentée dans le présent document permet à un envoyeur SCTP d'authentifier les tronçons en utilisant des clés partagées entre l'envoyeur et le receveur. Le receveur peut alors vérifier que les tronçons sont bien de l'envoyeur et non d'un attaquant malveillant (pour autant que l'attaquant ne connaît pas une clé partagée d'association).

L'extension décrite dans le présent document place le résultat du calcul d'un code d'authentification de message haché (HMAC, *Hashed Message Authentication Code*) avant les données couvertes par ce calcul. Le placer à la fin du paquet aurait exigé de placer un tronçon de contrôle après les tronçons de données en cas d'authentification de tronçons DATA. Cela enfreindrait la règle qui veut que les tronçons de contrôle soient produits avant les tronçons de données dans les paquets SCTP. Ont devrait aussi noter que mettre le résultat du calcul de HMAC après les données couvertes ne permettrait pas d'envoyer le paquet durant le calcul du HMAC parce que le résultat du calcul de HMAC est nécessaire pour calculer la somme de contrôle CRC32C du paquet SCTP, qui est placée dans l'en-tête commun du paquet SCTP.

L'extension SCTP pour la reconfiguration dynamique d'adresse requiert l'usage de l'extension décrite dans le présent document. L'extension de fiabilité partielle SCTP (PR-SCTP, *SCTP Partial Reliability Extension*) peut être utilisée conjointement à l'extension décrite dans le présent document.

2. Conventions

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 sont à interpréter comme décrit dans le BCP 14, [RFC2119].

3. Nouveaux types de paramètres

Cette Section définit les nouveaux types de paramètres qui vont être utilisés pour négocier l'authentification durant l'établissement d'association. Le Tableau 1 illustre ces nouveaux types de paramètres.

Type de paramètre	**Nom de paramètre**
0x8002	Aléatoire (RANDOM)
0x8003	Liste de tronçons (CHUNKS)
0x8004	Algorithme HMAC exigé (HMAC-ALGO)

Tableau 1

Noter que le format de paramètre exige que le receveur ignore le paramètre et continue le traitement si le paramètre n'est pas compris. Ceci est accompli (comme décrit au paragraphe 3.2.1 de la [RFC2960]) par l'utilisation des bits de poids fort du type de paramètre.

3.1 Paramètre aléatoire (RANDOM)

Ce paramètre est utilisé pour porter un nombre aléatoire de longueur arbitraire.

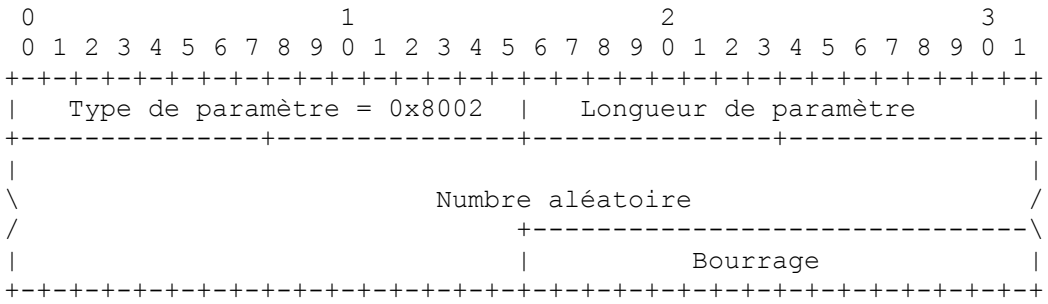


Figure 1

Type de paramètre : 2 octets (entier non signé) ; cette valeur DOIT être réglée à 0x8002.

Longueur de paramètre : 2 octets (entier non signé) ; cette valeur est la longueur du nomnre aléatoire en octets plus 4.

Nombre aléatoire : n octets (entier non signé) ; cette valeur représente un nombre aléatoire arbitraire dans l'ordre des octets du réseau.

Bourrage : 0, 1, 2, ou 3 octets (entier non signé). Si la longueur du nombre aléatoire n'est pas un multiple de 4 octets, l'envoyeur DOIT bourrer le paramètre avec des octets tout de zéros pour aligner le paramètre sur 32 bits. Le bourrage NE DOIT PAS être plus long que 3 octets et il DOIT être ignoré par le receveur.

Le paramètre RANDOM DOIT être inclus une fois dans le tronçon INIT ou INIT-ACK, si l'envoyeur veut envoyer ou recevoir des tronçons authentifiés, pour fournir un nombre aléatoire de 32 octets. Pour des nombres aléatoires de 32 octets, le bourrage est vide.

3.2 Paramètre liste de tronçons (CHUNKS)

Ce paramètre est utilisé pour spécifier de quels types de tronçons il est exigé qu'ils soient authentifiés avant d'être envoyés par l'homologue.

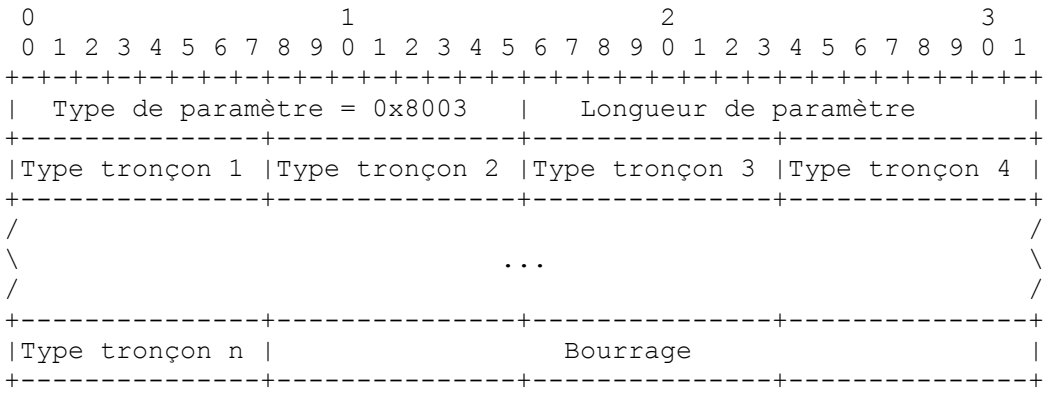


Figure 2

Type de paramètre : 2 octets (entier non signé). cette valeur DOIT être réglée à 0x8003.

Longueur de paramètre : 2 octets (entier non signé). cette valeur est le nombre de types de tronçons listés plus 4.

Type de tronçon n : 1 octet (entier non signé). Chaque type de tronçon mentionné doit être authentifié quand il est envoyé par l'homologue.

Bourrage : 0, 1, 2, ou 3 octets (entier non signé). Si la longueur du nombre aléatoire n'est pas un multiple de 4 octets, l'envoyeur DOIT bourrer le paramètre avec des octets tout de zéros pour aligner le paramètre sur 32 bits. Le bourrage NE DOIT PAS être plus long que 3 octets et il DOIT être ignoré par le receveur.

Le paramètre CHUNKS DOIT être inclus une fois dans le tronçon INIT ou INIT-ACK si l'envoyeur veut recevoir des tronçons authentifiés. Sa longueur maximum est 260 octets.

Les types de tronçons pour les tronçons INIT, INIT-ACK, SHUTDOWN-COMPLETE, et AUTH NE DOIVENT PAS être mentionnés dans le paramètre CHUNKS. Cependant, si un paramètre CHUNKS est reçu, alors les types pour les tronçons INIT, INIT-ACK, SHUTDOWN-COMPLETE, et AUTH DOIVENT être ignorés.

3.3 Paramètre Algorithme HMAC demandé (HMAC-ALGO)

Ce paramètre est utilisé pour faire la liste des identifiants HMAC que l'homologue DOIT utiliser.

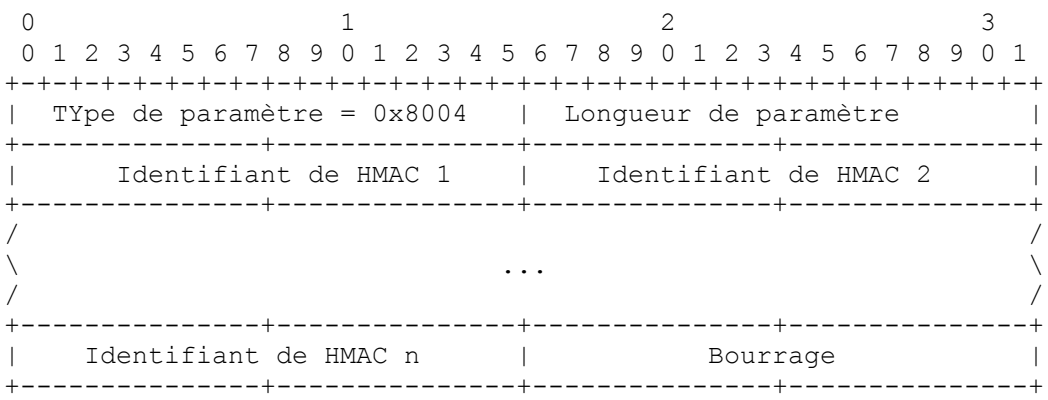


Figure 3

Type de paramètre : 2 octets (entier non signé). Cette valeur DOIT être réglée à 0x8004.

Longueur de paramètre : 2 octets (entier non signé). Cette valeur est le nombre d'identifiants HMAC multiplié par 2, plus 4.

Identifiant de HMAC n : 2 octets (entier non signé). Les valeurs exprimées sont une liste d'identifiants HMAC qui peuvent être utilisés par l'homologue. Les valeurs sont ordonnées par préférence, par rapport à l'envoyeur, où le premier Identifiant de HMACmentionné est celui qui est préféré par l'envoyeur.

Bourrage : 0 ou 2 octets (entier non signé). Si le nombre d'identifiants HMAC n'est pas pair, l'envoyeur DOIT bourrer le paramètre avec des octets tout à zéro pour aligner le paramètre sur 32 bits. Le bourrage DOIT être de 0 ou 2 octets et il DOIT être ignoré par le receveur.

Le paramètre HMAC-ALGO DOIT être inclus une fois dans le tronçon INIT ou INIT-ACK si l'envoyeur veut envoyer ou recevoir des tronçons authentifiés.

Le Tableau 2 montre les valeurs actuellement définies pour les identifiants de HMAC.

Identifiant de HMAC	Algorithme de résumé de message
0	Réservé
1	SHA-1 défini dans [FIPS180-2]
2	Réservé
3	SHA-256 défini dans [FIPS180-2]

Tableau 2

Chaque point d'extrémité qui prend en charge l'authentification de tronçon SCTP DOIT prendre en charge le HMAC sur la base de l'algorithme SHA-1.

4. Nouvelle cause d'erreur

Cette Section définit une nouvelle cause d'erreur qui va être envoyée si un tronçon AUTH est reçu avec un Identifiant de

HMAC non pris en charge. Le Tableau 3 illustre la nouvelle cause d'erreur.

Code de cause	Nom de cause d'erreur
0x0105	Identifiant de HMAC non pris en charge

Tableau 3

4.1 Cause d'erreur Identifiant HMAC non pris en charge

Cette cause d'erreur est utilisée pour indiquer qu'un tronçon AUTH a été reçu avec un identifiant de HMAC non pris en charge.

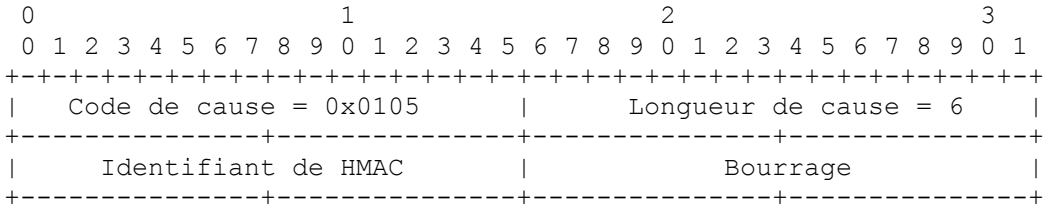


Figure 4

Code de cause : 2 octets (entier non signé). Cette valeur DOIT être réglée à 0x0105.

Longueur de cause : 2 octets (entier non signé). Cette valeur DOIT être réglée à 6.

Identifiant de HMAC : 2 octets (entier non signé). Cette valeur est l'identifiant de HMAC qui n'est pas supporté.

Bourrage : 2 octets (entier non signé). L'envoyeur DOIT bourrer la cause d'erreur avec des octets tout de zéros pour aligner la cause sur 32 bits. Le bourrage DOIT être de 2 octets et il DOIT être ignoré par le receveur.

5. Nouveau type de tronçon

Cette Section définit le nouveau type de tronçon qui va être utilisé pour authentifier les tronçons. Le Tableau 4 illustre le nouveau type de tronçon.

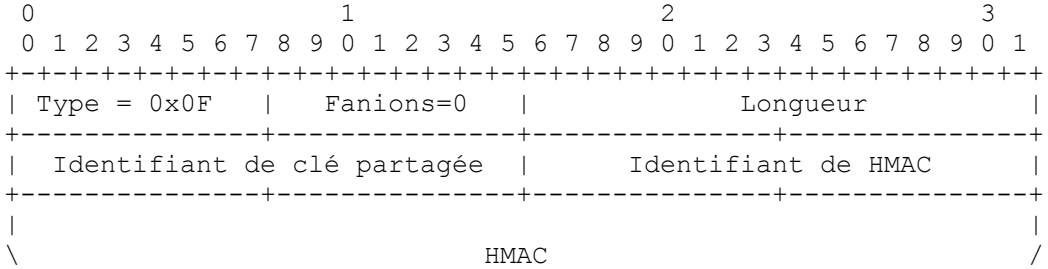
Type de tronçon	Nom de tronçon|
0x0F	Tronçon d'authentification (AUTH) |

Tableau 4

On devrait noter que le format de tronçon AUTH exige que le receveur ignore le tronçon si il n'est pas compris et qu'il élimine en silence tous les tronçons qui suivent. Ceci est accompli (comme décrit au paragraphe 3.2 de la [RFC2960]) par l'utilisation des bits de poids fort du type de tronçon.

5.1 Tronçon Authentification (AUTH)

Ce tronçon est utilisé pour contenir le résultat du calcul du HMAC.



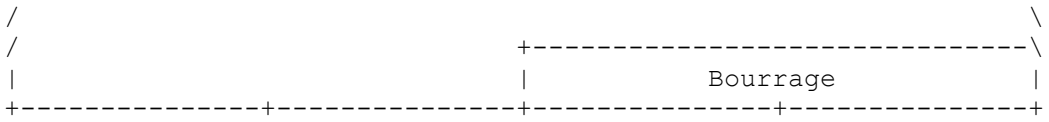


Figure 5

Type : 1 octet (entier non signé). Cette valeur DOIT être réglée à 0x0F pour tous les tronçons AUTH.

Fanions : 1 octet (entier non signé). DEVRAIT être réglé à zéro à l'émission et DOIT être ignoré à réception.

Longueur : 2 octets (entier non signé). Cette valeur contient la longueur du HMAC en octets plus 8.

Identifiant de clé partagée : 2 octets (entier non signé). Cette valeur décrit quelle clé partagée de paire de points d'extrémité est utilisée.

Identifiant de HMAC : 2 octets (entier non signé). Cette valeur décrit quel résumé de message est utilisé. Le Tableau 2 montre les valeurs actuellement définies.

HMAC : n octets (entier non signé). Cela contient le résultat du calcul du HMAC.

Bourrage : 0, 1, 2, ou 3 octets (entier non signé). Si la longueur du HMAC n'est pas un multiple de 4 octets, l'envoyeur DOIT bourrer le tronçon avec des octets tout à zéro pour faire que le tronçon soit aligné sur 32 bits. Le bourrage NE DOIT PAS faire plus de 3 octets et il DOIT être ignoré par le receveur.

Le tronçon de contrôle AUTH NE DOIT PAS apparaître plus d'une fois dans un paquet SCTP. Tous les tronçons de contrôle et de données qui sont placés après le tronçon AUTH dans le paquet sont envoyés authentifiés. Ces tronçons placés dans un paquet avant le tronçon AUTH ne sont pas authentifiés. Noter que les tronçons DATA ne peuvent pas apparaître avant les tronçons de contrôle dans un paquet SCTP.

6. Procédures

6.1 Établissement d'une clé partagée d'association

Un point d'extrémité SCTP qui veut recevoir ou envoyer des tronçons authentifiés DOIT envoyer un paramètre RANDOM dans son tronçon INIT ou INIT-ACK. Le paramètre RANDOM DOIT contenir un nombre aléatoire de 32 octets. Le nombre aléatoire devrait être généré conformément à la [RFC4086]. Si le nombre aléatoire n'est pas de 32 octets, l'association DOIT être interrompue. Le tronçon ABORT DEVRAIT contenir la cause d'erreur "Violation de protocole". Dans le cas d'une collision de INIT, les règles qui gouvernent le traitement de ce nombre aléatoire suivent le même schéma que pour l'étiquette de vérification, comme expliqué au paragraphe 5.2.4 de la [RFC2960]. Donc, chaque point d'extrémité connaît son propre nombre aléatoire et le nombre aléatoire de l'homologue après l'établissement de l'association.

Un point d'extrémité SCTP a une liste des tronçons qu'il accepte seulement si ils sont reçus authentifiés. Cette liste est incluse dans le INIT et INIT-ACK, et PEUT être omise si elle est vide.Comme cettte liste ne change pas durant la vie du point d'extrémité SCTP, il n'y a pas de problème en cas de collision de INIT.

Chaque point d'extrémité SCTP DOIT inclure dans le INIT et INIT-ACK un paramètre HMAC-ALGO contenant une liste d'identifiants de HMAC qu'il demande à l'homologue d'utiliser. Le receveur d'un paramètre HMAC-ALGO DEVRAIT utiliser le premier algorithme de la liste qu'il prend en charge. L'algorithme de HMAC fondé sur SHA-1 DOIT être pris en charge et inclus dans le paramètre HMAC-ALGO. Un point d'extrémité SCTP NE DOIT PAS changer les paramètres mentionnés dans le paramètre HMAC-ALGO durant la vie du point d'extrémité.

Les deux points d'extrémité d'une association PEUVENT avoir des clés partagées de paire de points d'extrémité qui sont des vecteurs d'octets et pré-configurées ou établies par un autre mécanisme. Elles sont identifiées par l'identifiant de clé partagée. Pour chaque clé partagée de paire de points d'extrémité, une clé partagée d'association est calculée. Si il n'y a pas de clé partagée de paire de points d'extrémité, seulement une clé partagée d'association est calculée en utilisant un vecteur d'octet vide comme clé partagée de paire de points d'extrémité.

Les paramètres RANDOM, CHUNKS, et HMAC-ALGO envoyés par chaque point d'extrémité sont enchaînés comme des vecteurs d'octets. Ces paramètres incluent le type de paramètre, la longueur de paramètre, et la valeur de paramètre, mais le

bourrage est omis ; tous le bourrage DOIT être supprimé de cet enchaînement avant de procéder au calcul des clés. Les paramètres qui n'ont pas été envoyés sont simplement omis du processus d'enchaînement. Les deux vecteurs résultant sont appelés les vecteurs de clé.

À partir des clés partagées de paire de points d'extrémité et des vecteurs de clé, les clés partagées d'association sont calculées. Ceci est effectué en choisissant le vecteur de clé le plus petit numériquement et en l'enchaînant à la clé partagée de la paire de points d'extrémité, et ensuite en enchaînant à cela le vecteur de clé numériquement plus grand. Si les vecteurs de clé sont égaux comme nombres mais diffèrent en longueur, alors l'ordre d'enchaînement est la clé partagée de point d'extrémité suivie par le plus court vecteur de clé, suivi par le plus long vecteur de clé. Autrement, si les vecteurs de clé sont identiques, ils peuvent être enchaînés dans n'importe quel ordre à la clé de paire de points d'extrémité. L'enchaînement est effectué sur les vecteurs d'octets, et toutes les comparaisons numériques utilisent l'ordre des octets du réseau pour convertir les vecteurs de clé en un nombre. Le résultat de l'enchaînement est la clé partagée d'association.

6.2 Envoi de tronçons authentifiés

Les points d'extrémité DOIVENT envoyer tous les tronçons demandés qui ont été authentifiés lorsque cela a été demandé par l'homologue. Les autres tronçons PEUVENT être envoyés qu'ils aient ou non été authentifiés. Si des clés partagées de paire de points d'extrémité sont utilisées, une d'elles DOIT être choisie pour l'authentification.

Pour envoyer des tronçons authentifiés, l'envoyeur DOIT inclure ces tronçons après un tronçon AUTH. Cela signifie qu'un envoyeur DOIT mettre en bouquet les tronçons afin de les authentifier.

Si le point d'extrémité n'a pas de clé partagée de paire de points d'extrémité pour l'homologue, il DOIT utiliser un identifiant de clé partagée de zéro avec une clé partagée de paire de points d'extrémité vide. Si il y a plusieurs clés partagées de point d'extrémité, l'envoyeur en choisit une et utilise l'identifiant de clé partagée correspondant.

L'envoyeur DOIT calculer le code d'authentification de message (MAC, *Message Authentication Code*) (comme décrit dans la [RFC2104]) en utilisant la fonction de hachage H décrite par l'identifiant de HMAC et la clé K d'association partagée sur la base de la clé partagée de paire de points d'extrémité décrite par l'identifiant de clé partagée. Les "données" utilisées pour le calcul du tronçon AUTH sont données par le tronçon AUTH avec son champ HMAC réglé à zéro (comme montré à la Figure 6) suivies par tous les tronçons qui sont placés après le tronçon AUTH dans le paquet SCTP.

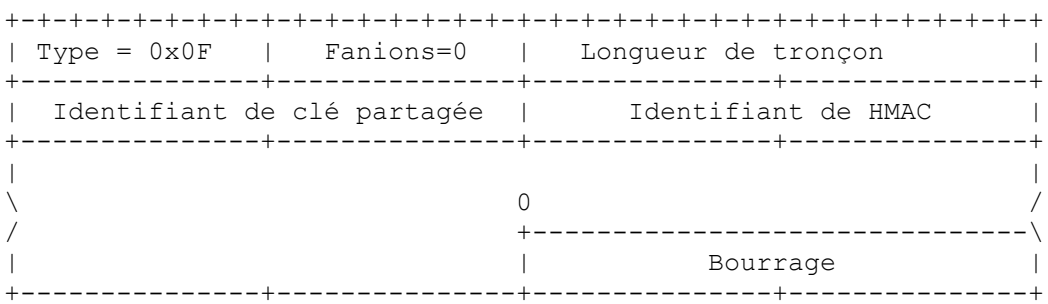


Figure 6

Noter que tous les champs sont dans l'ordre des octets du réseau et que le champ qui va contenir le HMAC complet est rempli de zéros. La longueur du champ montré comme zéro est la longueur du HMAC décrit par l'identifiant de HMAC. Le bourrage de tous les tronçons authentifiés DOIT être inclus dans le calcul de HMAC.

L'envoyeur met le HMAC dans le champ HMAC et envoie le paquet.

6.3 Réception de tronçons authentifiés

Le receveur a une liste des types de tronçons qu'il s'attend à recevoir seulement après un tronçon AUTH. Cette liste a été envoyée à l'homologue durant l'établissement de l'association. Il DOIT éliminer en silence ces tronçons si ils ne sont pas placés après un tronçon AUTH dans le paquet.

Le receveur DOIT utiliser l'algorithme de HMAC indiqué dans le champ Identifiant de HMAC. Si cet algorithme n'a pas été spécifié par le receveur dans le paramètre HMAC-ALGO dans le tronçon INIT ou INIT-ACK durant l'établissement de l'association, le tronçon AUTH et tous les tronçons après lui DOIVENT être éliminés et un tronçon ERROR DEVRAIT

être envoyé avec la cause d'erreur définie au paragraphe 4.1.

Si un point d'extrémité sans clé partagée reçoit un identifiant de clé partagée autre que 0, il DOIT éliminer en silence tous les tronçons authentifiés. Si le point d'extrémité a au moins une clé partagée de paire de points d'extrémité pour l'homologue, il DOIT utiliser la clé spécifiée par l'identifiant de clé partagée si une clé a été configurée pour cet identifiant de clé partagée. Si aucune clé partagée de paire de points d'extrémité n'a été configurée pour cet identifiant de clé partagée, tous les tronçons authentifiés DOIVENT être éliminés en silence.

Le receveur effectue maintenant le même calcul que décrit pour l'envoyeur sur la base de la Figure 6. Si le résultat du calcul est le même que celui donné dans le champ HMAC, tous les tronçons qui suivent le tronçon AUTH sont traités. Si le champ ne correspond pas au résultat du calcul, tous les tronçons qui suivent le tronçon AUTH DOIVENT être éliminés en silence.

On devrait noter que si le receveur veut supprimer une association d'une façon authentifiée seulement, le traitement des paquets mal formés ne devrait pas résulter en la suppression de l'association.

Un mise en œuvre de SCTP doit conserver l'état pour chaque association SCTP. Dans ce qui suit, on appelle cette structure de données le bloc de contrôle de transmission SCTP (STCB, *SCTP Transmission Control Block*).

Quand un point d'extrémité demande que les tronçons COOKIE-ECHO soient authentifiés, des procédures particulières doivent être suivies parce que la réception d'un tronçon COOKIE-ECHO pourrait résulter en la création d'une association SCTP. Si un paquet arrive en contenant un tronçon AUTH comme premier tronçon, un tronçon COOKIE-ECHO comme second tronçon, et éventuellement d'autres tronçons après eux, et si le receveur n'a pas de STCB pour ce paquet, alors l'authentification se fonde sur le contenu du tronçon COOKIE-ECHO. Dans cette situation, le receveur DOIT authentifier les tronçons dans le paquet en utilisant les paramètres RANDOM, les paramètres CHUNKS et les paramètres HMAC_ALGO obtenus du tronçon COOKIE-ECHO, et éventuellement un secret partagé local, comme entrées à la procédure d'authentification spécifiée au paragraphe 6.3. Si l'authentification échoue, alors le paquet est éliminé. Si l'authentification est réussie, le COOKIE-ECHO et tous les tronçons après le COOKIE-ECHO DOIVENT être traités. Si le receveur a un STCB, il DOIT traiter le tronçon AUTH comme décrit ci-dessus en utilisant le STCB provenant de l'association existante pour authentifier le tronçon COOKIE-ECHO et tous les tronçons après lui.

Si le receveur ne trouve pas de STCB pour un paquet contenant un tronçon AUTH comme premier tronçon et ne trouve pas de tronçon COOKIE-ECHO comme second tronçon, il DOIT utiliser les tronçons après le tronçon AUTH pour chercher une association existante. Si aucune association n'est trouvée, le paquet DOIT être considéré comme arrivé à l'improviste. Le traitement d'un paquet arrivé à l'improviste DOIT se faire sans prendre en compte le tronçon AUTH. Si une association est trouvée, il DOIT traiter le tronçon AUTH en utilisant le STCB provenant de l'association existante comme décrit précédemment.

Exiger que les tronçons ABORT et COOKIE-ECHO soient authentifiés rend impossible pour un attaquant de détruiure ou redémarrer une association tant que l'attaquant ne connait pas la clé partagée de l'association. Mais on devrait aussi noter que si un point d'extrémité n'accepte des tronçons ABORT que si ils sont authentifiés, il peut prendre plus longtemps pour détecter que l'homologue n'est plus disponible. Si un point d'extrémité accepte les tronçons COOKIE-ECHO seulement si ils sont authentifiés, la procédure de redémarrage ne fonctionne pas, parce que le point d'extrémité qui redémarre ne connaît probablement pas la clé partagée de l'association de la vieille association à redémarrer. Cependant, si le point d'extrémité qui redémarre connaît la vieille clé partagée d'association, il peut réussir à envoyer le tronçon COOKIE-ECHO d'une façon qui sera acceptée par l'homologue en utilisant cette vieille clé partagée d'association pour le paquet contenant le tronçon AUTH. Après cette opération, les deux points d'extrémité doivent utiliser la nouvelle clé partagée d'association.

Si un serveur a une clé partagée de paire de points d'extrémité avec certains clients, il peut demander que le tronçon COOKIE_ECHO soit authentifié et peut s'assurer que seules les associations provenant de clients avec une clé partagée de paire de points d'extrémité correcte sont acceptées.

De plus, il est important que le mouchard contenu dans un tronçon INIT-ACK et dans un tronçon COOKIE-ECHO NE CONTIENNE PAS de clé partagée de paire de points d'extrémité.

7. Exemples

Cette Section donne des exemples d'échanges de messages pour l'établissement d'association.

La façon la plus simple d'utiliser l'extension décrite dans le présent document est donnée par l'échange de messages suivant.

```
---------- INIT[RANDOM; CHUNKS; HMAC-ALGO] ---------->
<------- INIT-ACK[RANDOM; CHUNKS; HMAC-ALGO] ---------
-------------------- COOKIE-ECHO -------------------->
<-------------------- COOKIE-ACK ----------------------
```

Noter que le paramètre CHUNKS est facultatif dans le INIT et le INIT-ACK.

Si le serveur veut recevoir des tronçons DATA authentifiés, l'échange de messages suivant est possible :

```
---------- INIT[RANDOM; CHUNKS; HMAC-ALGO] ---------->
<------- INIT-ACK[RANDOM; CHUNKS; HMAC-ALGO] ---------
--------------- COOKIE-ECHO; AUTH; DATA ------------->
<----------------- COOKIE-ACK; SACK ------------------
```

Noter que si la clé partagée de paire de points d'extrémité dépend du client et du serveur, et est seulement connue de la couche supérieure, cet échange de messagee exige une intervention de la couche supérieure entre le traitement du tronçon COOKIE-ECHO et le traitement des tronçons AUTH et DATA du côté du serveur. Cette intervention peut être réalisée par une notification COMMUNICATION-UP suivie par la présentation de la clé partagée de paire de points d'extrémité par la couche supérieure à la pile SCTP, voir par exemple la Section 10 de la [RFC2960]. Si cette intervention n'est pas possible à cause de limitations de l'API (par exemple, la prise API) le serveur pourrait éliminier les tronçons AUTH et DATA, rendant nécessaire une retransmission du tronçon DATA. Si la même clé partagée de paire de points d'extrémité est utilisée pour plusieurs points d'extrémité et ne dépend pas du client, cette intervention pourrait n'être pas nécessaire.

8. Considérations relatives à l'IANA

Le présent document (RFC 4895) est la référence pour tous les enregistrements décrits dans cette section. Tous les enregistrements doivent être mentionnés dans le document disponible à SCTP-paramètres [SCTP-PARA]. Les changements sont décrits ci-dessous.

8.1 Nouveau type de tronçon

Un type de tronçon pour le tronçon AUTH a été alloué par l'IANA. IANA a alloué la valeur (15), comme indiqué au Tableau 4. Une ligne a été ajoutée au tableau "CHUNK TYPES" de SCTP-paramètres [SCTP-PARA]:

CHUNK TYPES

Valeur d'identifiant	**Type de tronçon**	**Référence**
15	Tronçon d'authentification (AUTH)	[RFC4895]

8.2 Trois nouveaux types de paramètres

Des types de paramètres ont été alloués par l'IANA pour les paramètres RANDOM, CHUNKS, et HMAC-ALGO. Les valeurs sont celles données au Tableau 1. Cela exige deux modifications aux tableaux "CHUNK PARAMETER TYPES" dans SCTP-paramètres [SCTP-PARA] : la première est l'ajout de trois nouvelles lignes au tableau "INIT Chunk Parameter Types":

Type de paramètre de tronçon	**Valeur**
Random	32770 (0x8002)
Liste de tronçons	32771 (0x8003)
Paramètre Algorithme HMAC demandé	32772 (0x8004)

Le second changement demandé est l'ajout des trois mêmes lignes au tableau "INIT ACK Chunk Parameter Types".

8.3 Nouvelle cause d'erreur

Une cause d'erreur de valeur 261 a été allouée pour Identifiant de HMAC non pris en charge (Tableau 3).

Cela exige une ligne supplémentaire du tableau "Codes de cause" dans les paramètres SCTP [SCTP-PARA] :

Valeur	Code de cause	Référence
261 (0x0105)	Identifiant de HMAC non pris en charge	[RFC4895]

8.4 Nouveau tableau d'identifiants HMAC

Les identifiants de HMAC doivent être conservés par l'IANA. Quatre valeurs initiales ont été allouées par l'IANA comme décrit au Tableau 2. Cela exige un nouveau tableau "Identifiants de HMAC" dans les paramètres SCTP [SCTP-PARA] :

Identifiant de HMAC	Algorithme de résumé de message	Référence
0	Réservé	[RFC4895]
1	SHA-1	[RFC4895]
2	Réservé	[RFC4895]
3	SHA-256	[RFC4895]

Pour enregistrer un nouvel identifiant de HMAC auprès de l'IANA dans ce tableau, il faut faire une demande d'allocation d'un numéro. Ce numéro doit être unique et un algorithme de résumé de message utilisable avec le HMAC défini dans la [RFC2104] DOIT être spécifié. La politique de "spécification exigée" de la [RFC2434] DOIT être appliquée.

9. Considérations sur la sécurité

Sans utiliser de clés partagées de point d'extrémité, cette extension protège seulement contre la modification ou l'injection de tronçons authentifiés par des attaquants qui n'ont pas capturé la prise de contact initiale d'établissement de l'association SCTP.

Si une clé partagée de paire de points d'extrémité est utilisée, même un vrai interposé ne peut pas injecter de tronçons, dont l'authentification est exigée, même si il intercepte l'échange initial de messages. Le point d'extrémité sait aussi qu'il accepte des tronçons authentifiés d'un homologue qui connaît la clé partagée de la paire de points d'extrémité.

L'établissement des clés partagées de paire de points d'extrémité sort du domaine d'application de ce document. D'autres mécanismes peuvent être utilisés, comme TLS ou une configuration manuelle.

Quand un point d'extrémité accepte les tronçons COOKIE-ECHO seulement quand ils sont authentifiés, la procédure de redémarrage ne fonctionne pas. Ni un attaquant ni un point d'extrémité redémarrant qui ne connaît pas la clé partagée d'association ne peut effectuer un redémarrage. Cependant, si la clé partagée d'association est connue, il est possible de redémarrer l'association.

Parce que SCTP a déjà un mécanisme incorporé qui traite la réception des tronçons dupliqués, la solution présentée utilise cette fonction et ne donne pas de méthode pour éviter par elle-même les attaques en répétition. Bien sûr, cela fonctionne seulement au sein de chaque association SCTP. Donc, une clé partagée séparée est utilisée pour chaque association SCTP pour traiter les attaques en répétition qui couvrent plusieurs associations SCTP.

Chaque point d'extrémité qui présente une liste de plus d'un élément dans le paramètre HMAC-ALGO doit être prêt à ce que l'homologue utilise le plus faible algorithme de la liste.

Quand une paire de points d'extrémité utilise des clés partagées de paire de points d'extrémité non NULLES et qu'un des points d'extrémité accepte quand même une clé NULLE, un attaquant qui a capturé la prise de contact initiale peut alors injecter ou modifier des tronçons authentifiés en utilisant la clé NULLE.

10. Remerciements

Les auteurs remercient David Black, Sascha Grau, Russ Housley, Ivan Arias Rodriguez, Irene Ruengeler, et Magnus

Westerlund de leurs précieux commentaires.

11. Références normatives

[RFC1321] R. Rivest, "Algorithme de résumé de message MD5", avril 1992. *(Information)*

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

[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)

[RFC2960] R. Stewart et autres, "Protocole de transmission de commandes de flux", octobre 2000. (*Obsolète, voir* RFC4960) *(P.S.)*

[RFC3436] A. Jungmaier, E. Rescorla, M. Tuexen, "Sécurité de la couche Transport sur le protocole de transmission de contrôle de flux", décembre 2002. *(P.S.)*

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

[FIPS180-2] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, août 2002, <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf >.

[SCTP-PARA] <http://www.iana.org/assignments/sctp-parameters >

Adresse des auteurs

Michael Tuexen
Muenster Univ. of Applied Sciences
Stegerwaldstr. 39
48565 Steinfurt
Germany
mél : tuexen@fh-muenster.de

Randall R. Stewart
Cisco Systems, Inc.
4875 Forest Drive
Columbia, SC 29206
USA
mél : rrs@cisco.com

Peter Lei
Cisco Systems, Inc.
8735 West Higgins Road
Suite 300
Chicago, IL 60631
USA
mél : peterlei@cisco.com

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA
mél : ekr@rtfm.com

Déclaration complète de droits de reproduction

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