Groupe de travail Réseau
Request for Comments : 4121
RFC mise à jour : 1964
Mise à jour par RFC6112, RFC6542, RFC6649, RFC8062
Catégorie : En cours de normalisation

L. Zhu & K. Jaganathan, Microsoft
S. Hartman, MIT
juillet 2005

Traduction Claude Brière de L'Isle

Version 2 du mécanisme d'interface de programme d'application de service de sécurité générique (GSS-API) de Kerberos version 5

Résumé
Le présent document définit les protocoles, procédures, et conventions à employer par les homologues qui mettent en œuvre le service générique de sécurité d'interface de programme d'application (GSS-API, *Generic Security Service Application Program Interface*) lors de l'utilisation du mécanisme Kerberos version 5.

La RFC 1964 est mise à jour et des changements incrémentaires sont proposés en réponse à des développements récents comme l'introduction du cadre de cryptosystème Kerberos. Ces changements supportent l'inclusion de nouveaux cryptosystèmes, en définissant de nouveaux jetons par message ainsi que leurs algorithmes de chiffrement et de somme de contrôle sur la base des profils de cryptosystèmes.

Table des matières

1. Introduction

La [RFC3961] définit un cadre générique pour décrire les types de chiffrement et de somme de contrôle à utiliser avec le protocole Kerberos et les protocoles associés.

La [RFC1964] décrit le mécanisme GSS-API pour Kerberos version 5. Elle définit le format de l'établissement de contexte, les jetons par message et de suppression de contexte, et utilise des identifiants d'algorithme pour chaque cryptosystème dans les jetons par message et de suppression de contexte.

L'approche retenue par le présent document pallie le besoin d'identifiants d'algorithmes. Cela est réalisé par le même algorithme de chiffrement, spécifié par le profil de chiffrement [RFC3961] pour la clé ou sous clé de session qui est créée durant la négociation de contexte, et son algorithme de somme de contrôle obligatoire. La disposition de message des jetons par message est donc révisée pour supprimer les indicateurs d'algorithme et pour ajouter des informations supplémentaires pour prendre en charge le cadre de chiffrement générique [RFC3961].

Les jetons transférés entre les homologues GSS-API pour l'établissement du contexte de sécurité sont aussi décrits dans le présent document. Les éléments de données échangés entre une mise en œuvre de point d'extrémité GSS-API et le centre de distribution de clés (KDC, *Key Distribution Center*) Kerberos [RFC4120] ne sont pas spécifiques de l'usage de GSS-API et sont donc définis dans la [RFC4120] plutôt que dans la présente spécification.

Les nouveaux formats de jetons spécifiés dans le présent document DOIVENT être utilisés avec tous les "nouveaux" types de chiffrement [RFC4120] et PEUVENT être utilisés avec des types de chiffrement qui ne sont pas "nouveaux", pourvu que l'initiateur et l'accepteur sachent d'après l'établissement de contexte qu'ils peuvent tous deux traiter ces nouveaux formats de jetons.

Les types de chiffrement "nouveaux" sont ceux qui ont été spécifiés avec ou depuis la spécification du nouveau cryptosystème Kerberos [RFC3961], comme défini au paragraphe 3.1.3 de la [RFC4120]. La liste des types de chiffrement qui ne sont pas nouveaux est la suivante [RFC3961] :

Type de chiffrement	**Numéro alloué**
des-cbc-crc	1
des-cbc-md4	2
des-cbc-md5	3
des3-cbc-md5	5
des3-cbc-sha1	7
dsaWithSHA1-CmsOID	9
md5WithRSAEncryption-CmsOID	10
sha1WithRSAEncryption-CmsOID	11
rc2CBC-EnvOID	12
rsaEncryption-EnvOID	13
rsaES-OAEP-ENV-OID	14
des-ede3-cbc-Env-OID	15
des3-cbc-sha1-kd	16
rc4-hmac	23

Conventions utilisées dans le présent document

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].

Le terme "ordre petit boutien" est utilisé comme abrégé pour se référer au codage avec l'octet de moindre poids en premier, tandis que le terme "ordre gros boutien" est pour le codage avec l'octet de plus fort poids en premier.

2. Déduction de clé pour les jetons par message

Pour limiter l'exposition d'une certaine clé, la [RFC3961] a adopté les clés déduites "unidirectionnelles" à "préservation d'entropie", à partir d'une clé de base ou clé de protocole, pour différents objets ou usages de clé.

Le présent document définit ci-dessous quatre valeurs d'usage de clés qui sont utilisées pour déduire une clé spécifique pour signer et sceller les messages à partir de la clé ou sous clé de session [RFC4120] créée durant l'établissement du contexte.

Nom	**Valeur**
KG-USAGE-ACCEPTOR-SEAL	22
KG-USAGE-ACCEPTOR-SIGN	23
KG-USAGE-INITIATOR-SEAL	24
KG-USAGE-INITIATOR-SIGN	25

Lorsque l'envoyeur est celui qui accepte le contexte, la valeur KG-USAGE-ACCEPTOR-SIGN est utilisée comme numéro d'usage dans la fonction de déduction de clé pour déduire les clés à utiliser dans les jetons MIC (comme défini au paragraphe 4.2.6.1). KG-USAGE-ACCEPTOR-SEAL est utilisée pour les jetons Wrap (comme défini au paragraphe 4.2.6.2). De même, lorsque l'envoyeur est l'initiateur du contexte, KG-USAGE-INITIATOR-SIGN est utilisée comme numéro d'usage dans la fonction de déduction de clé pour les jetons MIC, tandis que KG-USAGE-INITIATOR-SEAL est utilisée pour les jetons Wrap. Même si le jeton Wrap n'assure pas la confidentialité, les mêmes valeurs d'usage que spécifiées ci-dessus sont utilisées.

Durant la séquence d'initiation et d'acceptation de contexte, l'acceptant PEUT faire valoir une sous clé dans le message AP-REP. Si l'acceptant fait valoir une sous clé, la clé de base est la sous clé que l'acceptant a fait valoir et les jetons par message suivants DOIVENT être marqués du fanion "AcceptorSubkey", comme décrit au paragraphe 4.2.2. Autrement, si l'initiateur fait valoir une sous clé dans le message AP-REQ, la clé de base est cette sous clé ; si l'initiateur ne fait pas valoir de sous clé, la clé de base est la clé de session dans le ticket de service.

3. Qualité de protection

La spécification GSS-API [RFC2743] donne les valeurs de qualité de protection (QOP) qui peuvent être utilisées par les applications pour demander un certain type de chiffrement ou signature. Une valeur de QOP de zéro est utilisée pour indiquer la protection par "défaut" ; les applications qui n'utilisent par la QOP par défaut n'ont pas la garantie d'être portables entre les mises en œuvre, ou même d'interopérer avec les différents déploiements de configurations de la même mise en œuvre. Utiliser un algorithme différent de celui pour lequel la clé est définie peut n'être pas approprié. Donc, lorsque la nouvelle méthode du présent document est utilisée, la valeur de QOP est ignorée.

Les algorithmes de chiffrement et de somme de contrôle dans les jetons par message sont maintenant implicitement définis par les algorithmes associés à la clé ou sous clé de session. Donc, les identifiants d'algorithme décrits dans la [RFC1964] ne sont plus nécessaires et sont retirés des nouveaux en-têtes de jetons.

4. Définitions et formats de jetons

Cette section fournit les termes et définitions, ainsi que les descriptions des jetons spécifiques du mécanisme GSS-API de Kerberos version 5.

4.1 Jetons d'établissement de contexte

Tous les jetons d'établissement de contexte émis par le mécanismes GSS-API de Kerberos version 5 DEVRONT avoir le tramage décrit au paragraphe 3.1 de la [RFC2743], comme illustré par les structures pseudo-ASN.1 suivantes :

```
DÉFINITIONS GSS-API ::=

DÉBUT

MechType ::= IDENTIFIANT D'OBJET        -- représente le mécanisme Kerberos V5

GSSAPI-Token ::=        -- indication d'option (délégation, etc.) indiquée au sein du jeton spécifique du mécanisme
      [APPLICATION 0] IMPLICIT SEQUENCE {
            thisMech MechType,
            innerToken ANY DEFINED BY thisMech  -- structure ASN.1 spécifique du contenu du mécanisme non exigée
            }
FIN
```

Le champ innerToken commence par un identifiant de jeton de deux octets (TOK_ID) exprimé en ordre gros boutien, suivi par un message Kerberos.

Suivent les valeurs de TOK_ID utilisées dans les jetons d'établissement de contexte :

Jeton TOK_ID	**Valeur en Hex**
KRB_AP_REQ	01 00
KRB_AP_REP	02 00
KRB_ERROR	03 00

Les messages Kerberos KRB_AP_REQUEST, KRB_AP_REPLY, et KRB_ERROR sont définis dans la [RFC4120].

Si un identifiant de jeton (TOK_ID) inconnu est reçu dans le jeton d'établissement de contexte initial, le receveur DOIT retourner l'état majeur GSS_S_CONTINUE_NEEDED, et le jeton de résultat retourné DOIT contenir un message KRB_ERROR avec le code d'erreur KRB_AP_ERR_MSG_TYPE [RFC4120].

4.1.1 Somme de contrôle d'authentificateur

L'authentificateur dans le message KRB_AP_REQ DOIT inclure le numéro de séquence facultatif et le champ Somme de contrôle. Le champ Somme de contrôle est utilisé pour porter les fanions de service, les liens de canaux, et les informations de délégation facultatives.

Le type de somme de contrôle DOIT être 0x8003. Lorsque la délégation est utilisée, un ticket d'accord de ticket sera transféré dans un message KRB_CRED. Ce ticket DEVRAIT avoir son fanion Transmissible établi. Le champ EncryptedData du message KRB_CRED [RFC4120] DOIT être chiffré dans la clé de session du ticket utilisé pour authentifier le contexte.

Le champ de somme de contrôle de l'authentificateur DEVRA avoir le format suivant :

Octet	Nom	Description
0..3	Lgth	Nombre d'octets dans le champ Bnd ; représenté en ordre petit boutien ; contient la valeur hex 10 00 00 00 (16).
4..19	Bnd	Informations de lien de canal, comme décrit au paragraphe 4.1.1.2.
20..23	Flags	Fanions d'établissement de contexte de quatre octets en ordre petit boutien comme décrit au paragraphe 4.1.1.1.
24..25	DlgOpt	Identifiant d'option de délégation (=1) en ordre petit boutien [facultatif]. Ce champ et les deux suivants sont présents si et seulement si GSS_C_DELEG_FLAG est établi comme décrit au paragraphe 4.1.1.1.
26..27	Dlgth	Longueur du champ Deleg en ordre petit boutien [facultatif].
28..(n-1)	Deleg	Message KRB_CRED (n = Dlgth + 28) [facultatif].
n..fin	Exts	Extensions [facultatif].

La longueur du champ Somme de contrôle DOIT être d'au moins 24 octets quand GSS_C_DELEG_FLAG n'est pas établi (comme décrit au paragraphe 4.1.1.1), et d'au moins 28 octets plus Dlgth octets quand GSS_C_DELEG_FLAG est établi. Lorsque GSS_C_DELEG_FLAG est établi, les champs DlgOpt, Dlgth, et Deleg des données de la somme de contrôle DOIVENT immédiatement suivre le champ Flags. Les octets facultatifs en queue (à savoir le champ "Exts") facilitent les futures extensions de ce mécanisme. Lorsque on n'utilise pas la délégation, mais que le champ Exts est présent, le champ Exts commence à l'octet 24 (DlgOpt, Dlgth et Deleg sont absents).

Les initiateurs qui ne prennent pas en charge les extensions NE DOIVENT PAS inclure plus de 24 octets dans le champ Somme de contrôle (lorsque GSS_C_DELEG_FLAG n'est pas établi) ou plus de 28 octets plus le KRB_CRED dans le champ Deleg (lorsque GSS_C_DELEG_FLAG est établi). Les accepteurs qui ne comprennent pas les extensions DOIVENT ignorer tous les octets après le champ Deleg des données de la somme de contrôle (lorsque GSS_C_DELEG_FLAG est établi) ou après le champ Flags des données de la somme de contrôle (quand GSS_C_DELEG_FLAG n'est pas établi).

4.1.1.1 Champ de fanions de somme de contrôle

Le champ "Flags" de somme de contrôle est utilisé pour porter les options de service ou les informations de négociation d'extensions.

Les fanions d'établissement de contexte suivants sont définis dans la [RFC2744].

Nom de fanion	Valeur
GSS_C_DELEG_FLAG	1
GSS_C_MUTUAL_FLAG	2
GSS_C_REPLAY_FLAG	4
GSS_C_SEQUENCE_FLAG	8
GSS_C_CONF_FLAG	16
GSS_C_INTEG_FLAG	32

Les fanions d'établissement de contexte sont exposés à l'application appelante. Si l'application appelante désire une option de service particulière, elle demande alors cette option via GSS_Init_sec_context() [RFC2743]. Si les valeurs d'état de retour correspondantes [RFC2743] indiquent que tous les services facultatifs de niveau contexte seront actifs sur le contexte, les valeurs de fanion correspondantes dans le tableau ci-dessus DOIVENT être établies dans le champ Fanions de somme de contrôle.

Les valeurs de fanion 4 096 à 524 288 (2^12, 2^13, ..., 2^19) sont réservées à l'usage des extensions de ce mécanisme spécifiques des fabricants traditionnels.

Toutes les autres valeurs de fanion non spécifiées ici sont réservées pour une utilisation future. De futures révisions de ce mécanisme POURRONT utiliser ces fanions réservés et POURRONT s'appuyer sur les mises en œuvre de cette version pour ne pas utiliser de tels fanions afin de négocier correctement les versions de mécanisme. Les valeurs de fanion non définies DOIVENT être mises à zéro par l'envoyeur et les fanions inconnus DOIVENT être ignorés par le receveur.

4.1.1.2 Informations de lien de canal

Ces étiquettes sont destinées à être utilisées pour identifier le canal de communications particulier pour lequel sont destinés les jetons d'établissement de contexte de sécurité GSS-API, limitant donc la portée dans laquelle un jeton d'établissement de contexte intercepté peut être réutilisé par un attaquant (voir au paragraphe1.1.6 de la [RFC2743]).

Lorsque on utilise des liens de langage C, les liens de canal sont communiqués à la GSS-API en utilisant la structure suivante [RFC2744] :

```
typedef struct gss_channel_bindings_struct {
      OM_uint32       initiator_addrtype; gss_buffer_desc initiator_address;
      OM_uint32       acceptor_addrtype; gss_buffer_desc acceptor_address; gss_buffer_desc application_data;
    } *gss_channel_bindings_t;
```

Les champs et constantes membres utilisés pour les différents types d'adresse sont définis dans la [RFC2744].

Le champ "Bnd" contient le hachage MD5 des liens de canal, pris sur tous les composants non nuls des liens, dans l'ordre de déclaration. Les champs d'entiers au sein des liens de canal sont représentés en ordre petit boutien pour les besoins du calcul MD5.

Dans le calcul du contenu du champ Bnd, les point de détail suivants s'appliquent :

(1) Pour le calcul du hachage MD5, chaque champ d'entier et champ de longueur d'entrée DEVRA être formaté sur quatre octets, en utilisant l'ordre d'octets petit boutien.

(2) Tous les champs de longueur d'entrée au sein de l'élément gss_buffer_desc d'une gss_channel_bindings_struct, même ceux qui sont de valeur zéro, DEVRONT être inclus dans le calcul du hachage. Les éléments de valeur des éléments gss_buffer_desc DEVRONT être déréférencés, et les données résultantes DEVRONT être incluses dans le calcul du hachage, seulement dans le cas des éléments gss_buffer_desc qui ont des spécificateurs de longueur non zéro.

(3) Si l'appelant passe la valeur GSS_C_NO_BINDINGS au lieu d'une structure valide de lien de canal, le champ Bnd DEVRA être réglé à 16 octets de valeur zéro.

Si l'appelant à GSS_Accept_sec_context [RFC2743] passe un GSS_C_NO_CHANNEL_BINDINGS [RFC2744] comme lien de canal, l'acceptant PEUT alors ignorer tous les liens de canal fournis par l'initiateur, en retournant un succès même si l'initiateur a bien passé les liens de canal.

Si l'application fournit, dans les liens de canal, une mémoire tampon avec un champ de longueur de plus de 4 294 967 295 (2^32 - 1) la mise en œuvre de ce mécanisme PEUT choisir de rejeter tous les liens de canal, en utilisant l'état majeur GSS_S_BAD_BINDINGS [RFC2743]. Dans tous les cas, la taille de la mémoire tampon des données de lien de canal qui peut être utilisée (interopérable, sans extension) avec la présente spécification est limitée à 4 294 967 295 octets.

4.2 Jetons par message

Deux classes de jetons sont définies dans ce paragraphe : (1) des jetons "MIC", émis par des appels à GSS_GetMIC() et consommés par des appels à GSS_VerifyMIC(), et (2) des jetons "Wrap", émis par des appels à GSS_Wrap() et consommés par des appels à GSS_Unwrap().

Ces nouveaux jetons par message n'incluent pas le tramage de jeton générique GSS-API utilisé par les jetons d'établissement de contexte. Ces nouveaux jetons sont conçus pour être utilisés avec les nouveaux cryptosystèmes qui peuvent avoir des sommes de contrôle de taille variable.

4.2.1 Numéro de séquence

Pour distinguer les messages répétés intentionnellement de ceux répétés de façon malveillante, les jetons par message contiennent un champ numéro de séquence, qui est un entier de 64 bits exprimé en ordre gros boutien. Après l'envoi d'un jeton GSS_GetMIC() ou GSS_Wrap(), les numéros de séquence de l'envoyeur DEVRONT être incrémentés de un.

4.2.2 Champ Flags

Le champ "Flags" est un entier de un octet utilisé pour indiquer un ensemble d'attributs pour le message protégé. Par exemple, un fanion est alloué comme indicateur de direction, empêchant ainsi l'acceptation du même message renvoyé dans la direction inverse par un adversaire.

La signification des bits dans ce champ (le bit de moindre poids est le bit 0) est la suivante :

Bit	Nom	Description
0	SentByAcceptor	Mis à 1, ce fanion indique que l'envoyeur est celui qui accepte le contexte. Mis à zéro, il indique que l'envoyeur est l'initiateur du contexte.
1	Sealed	Établi dans les jetons Wrap, ce fanion indique que la confidentialité est fournie. Il NE DEVRA PAS être établi dans les jetons MIC.
2	AcceptorSubkey	Une sous clé affirmée par l'accepteur du contexte est utilisée pour protéger le message.

Le reste des bits disponibles est réservé pour une future utilisation et ils DOIVENT être à zéro. Le receveur DOIT ignorer les fanions inconnus.

4.2.3 Champ EC

Le champ "EC" (Extra Count, *compte supplémentaire*) est un champ d'entier de deux octets exprimé en ordre gros boutien.

Dans les jetons Wrap avec confidentialité, le champ EC DEVRA être utilisé pour coder le nombre d'octets dans le remplissage, comme décrit au paragraphe 4.2.4.

Dans les jetons Wrap sans confidentialité, le champ EC DEVRA être utilisé pour coder le nombre d'octets dans la somme de contrôle en queue, comme décrit au paragraphe 4.2.4.

4.2.4 Opérations de chiffrement et de somme de contrôle

Les algorithmes de chiffrement définis par les profils de chiffrement assurent la protection de l'intégrité [RFC3961]. Donc, aucune somme de contrôle séparée n'est nécessaire.

Le résultat du déchiffrement peut être plus long que le texte source original [RFC3961] et les octets de remplissage supplémentaires sont appelés le "résidu du cryptosystème" dans ce document. Cependant, étant donnée la taille de toutes données de texte source, on peut toujours trouver une taille (éventuellement supérieure) telle que quand on bourre le texte à chiffrer jusqu'à cette taille, il n'y ait pas de résidu de cryptosystème à ajouter [RFC3961].

Dans les jetons Wrap qui assurent la confidentialité, les 16 premiers octets du jeton Wrap ("l'en-tête", comme défini au paragraphe 4.2.6) DEVRONT être ajoutés aux données du texte source avant le chiffrement. Les octets de remplissage PEUVENT être insérés entre les données du texte source et "l'en-tête". Les valeurs et tailles des octets de remplissage sont choisies par la mise en œuvre, de telle sorte qu'il NE DEVRA PAS y avoir de résidu de cryptosystème présent après le déchiffrement. Le jeton Wrap résultant est {"en-tête" | (données de texte source) chiffrées | remplissage | "en-tête")}, où chiffrées est l'opération de chiffrement (qui assure la protection de l'intégrité) définie dans le profil de chiffrement [RFC3961], et le champ RRC (comme défini au paragraphe 4.2.5) dans l'en-tête à chiffrer contient la valeur hexadécimale 00 00.

Dans les jetons Wrap qui n'assurent pas la confidentialité, la somme de contrôle DEVRA être calculée d'abord sur les données du texte source à signer, et ensuite sur les 16 premiers octets du jeton Wrap (l'"en-tête", comme défini au paragraphe 4.2.6). Le champ EC et le champ RRC dans l'en-tête de jeton DEVRONT être remplis de zéros pour les besoins du calcul de la somme de contrôle. Le jeton Wrap résultant est {"en-tête" | données du texte source | get_mic(données du

texte source | "en-tête")}, où get_mic() est l'opération de somme de contrôle pour le mécanisme de somme de contrôle requis du mécanisme de chiffrement choisi défini dans le profil cryptographique [RFC3961].

Les paramètres pour la clé et l'état de chiffrement dans les opérations encrypt() et get_mic() ont été omis pour faire court.

Pour les jetons MIC, la somme de contrôle DEVRA être calculée comme suit : l'opération de somme de contrôle est d'abord calculée sur les données du texte source à signer, et ensuite sur les 16 premiers octets du jeton MIC, où le mécanisme de somme de contrôle est le mécanisme de somme de contrôle requis du mécanisme de chiffrement choisi défini dans le profil de chiffrement [RFC3961].

Les jetons Wrap et MIC résultants lient les données à l'en-tête de jeton, incluant le numéro de séquence et l'indicateur de direction.

4.2.5 Champ RRC

Le champ "RRC" (Right Rotation Count, *compte de rotation à droite*) dans les jetons Wrap est ajouté pour permettre aux données d'être chiffrées en place par les applications existantes d'interface de fournisseur de service de sécurité (SSPI, *Security Service Provider Interface*) [SSPI] qui ne fournissent pas de mémoire tampon supplémentaire pour l'en queue (le texte chiffré après les données chiffrées en place) en plus de la mémoire tampon pour l'en-tête (le texte chiffré avant les données chiffrées en place). En excluant les 16 premiers octets de l'en-tête de jeton, le jeton Wrap résultant dans le paragraphe précédent subit une rotation à droite de "RRC" octets. Le résultat net est que "RRC" octets des octets de queue sont déplacés vers l'en-tête.

Considérons ce qui suit comme un exemple de cette opération de rotation: Supposons que la valeur de RRC soit 3 et que le jeton avant la rotation soit {"en-tête" | aa | bb | cc | dd | ee | ff | gg | hh}. Le jeton après rotation va être {"en-tête" | ff | gg | hh | aa | bb | cc | dd | ee }, où {aa | bb | cc |...| hh} sera utilisé pour indiquer la séquence d'octets.

Le champ RRC est exprimé comme un entier de deux octets en ordre gros boutien.

La valeur du compte de rotation est choisie par l'envoyeur sur la base des détails de la mise en œuvre. Le receveur DOIT être capable d'interpréter toutes les valeurs de compte de rotation possibles, incluant des comptes de rotation supérieurs à la longueur du jeton.

4.2.6 Présentations de message

Les jetons par message commencent par un champ d'identifiant de jeton de deux octets (TOK_ID) exprimé en ordre gros boutien. Ces jetons sont définis séparément dans les paragraphes suivants.

4.2.6.1 Jetons MIC

L'utilisation de l'invocation de GSS_GetMIC() donne un jeton (appelé le jeton MIC dans ce document) séparé des données d'utilisateur qui sont protégées, qui peut être utilisé pour vérifier l'intégrité de ces données à réception. Le jeton a le format suivant

N° d'octet	Nom	Description
0..1	TOK_ID	Champ d'identification. Les jetons émis par GSS_GetMIC() contiennent la valeur hex de 04 04 exprimée en ordre gros boutien dans ce champ.
2	Flags	Champ d'attributs, comme décrit au paragraphe 4.2.2.
3..7	Filler	Contient cinq octets de valeur hex FF.
8..15	SND_SEQ	Champ Numéro de séquence en clair, exprimé en ordre gros boutien.
16..fint	SGN_CKSUM	Somme de contrôle des données "à signer" et les octets 0 à 15, comme décrit au § 4.2.4.

Le champ de remplissage est inclus dans le calcul de la somme de contrôle pour simplifier.

4.2.6.2 Jetons Wrap

L'utilisation de l'invocation de GSS_Wrap() donne un jeton (appelé jeton Wrap dans ce document) qui consiste en un en-tête descriptif, suivi par une portion de corps qui contient l'entrée des données d'utilisateur en texte source enchaîné avec la somme de contrôle, ou l'entrée des données d'utilisateur chiffrées. Le jeton GSS_Wrap() DEVRA avoir le format suivant :

N° d'octet	Nom	Description
0..1	TOK_ID	Champ d'identification. Les jetons émis par GSS_Wrap() contiennent la valeur hex de 05 04 exprimée en ordre gros boutien dans ce champ.
2	Flags	Champ d'attributs, comme décrit au paragraphe 4.2.2.
3	Filler	Contient la valeur hexadécimale FF.
4..5	EC	Contient le champ "extra count", en ordre gros boutien comme décrit au paragraphe 4.2.3.
6..7	RRC	Contient le "compte de rotation droite" en ordre gros boutien, comme décrit au § 4.2.5.
8..15	SND_SEQ	Champ Numéro de séquence en clair, exprimé en ordre gros boutien.
16...fin	Data	Données chiffrées pour jetons Wrap avec confidentialité, ou données de texte source suivies par la somme de contrôle pour les jetons Wrap sans confidentialité, comme décrit au paragraphe 4.2.4.

4.3 Jetons de suppression de contexte

Les jetons de suppression de contexte sont vides dans ce mécanisme. Les deux homologues d'un contexte de sécurité invoquent GSS_Delete_sec_context() [RFC2743] indépendamment, passant une mémoire tampon output_context_token nulle pour indique qu'aucun context_token n'est requis. Les mises en œuvre de GSS_Delete_sec_context() devraient supprimer les informations de contexte pertinentes mémorisées localement.

4.4 Considérations d'allocation d'identifiant de jeton

Les identifiants de jeton (TOK_ID) de 0x60 0x00 à 0x60 0xFF inclus sont réservés et NE DEVRONT pas être alloués. Donc, en examinant les deux premiers octets d'un jeton, on peut dire sans ambiguïté si il est enveloppé avec le tramage de jeton générique GSS-API.

5. Définitions de paramètres

Cette section définit les valeurs de paramètres utilisées par le mécanisme GSS-API Kerberos V5. Elle définit les éléments d'interface qui prennent en charge la portabilité, et suppose l'utilisation de liens de langage C selon la [RFC2744].

5.1 Codes d'état mineurs

Ce paragraphe recommande des noms symboliques communs pour les valeurs d'état mineur à retourner par le mécanisme GSS-API Kerberos V5. L'utilisation de ces définitions permettra à des mises en œuvre indépendantes d'améliorer la portabilité d'application à travers des mises en œuvre différentes du mécanisme défini dans cette spécification. (Dans tous les cas, la mise en œuvre de GSS_Display_status() permettra aux appelants de convertir les indicateurs minor_status en représentations de texte.) Chaque mise en œuvre devrait rendre disponible, par des fichiers inclus ou d'autres moyens, une facilité de traduire ces noms symboliques en les valeurs concrètes qu'utilise une mise en œuvre GSS-API particulière pour représenter les valeurs de minor_status spécifiées dans cette section.

La liste PEUT croître avec le temps et le besoin de codes d'états mineurs supplémentaires, spécifiques de mises en œuvre particulières, PEUT survenir. Cependant, il est recommandé que les mises en œuvre retournent une valeur de minor_status comme défini sur la base du mécanisme au sein de cette section quand ce code représente précisément un état rapportable plutôt que d'utiliser un code séparé, défini par la mise en œuvre.

5.1.1 Codes non spécifiques de Kerberos

```
GSS_KRB5_S_G_BAD_SERVICE_NAME   /* "Pas de @ dans la chaîne de nom SERVICE-NAME " */
GSS_KRB5_S_G_BAD_STRING_UID     /* "STRING-UID-NAME contient des non chiffres" */
GSS_KRB5_S_G_NOUSER             /* "UID ne se résout pas en nom d'utilisateur" */
GSS_KRB5_S_G_VALIDATE_FAILED    /* "Erreur de validation" */
GSS_KRB5_S_G_BUFFER_ALLOC       /* "Les données de gss_buffer_t n'ont pas pu être allouées" */
GSS_KRB5_S_G_BAD_MSG_CTX        /* "Contexte de message invalide" */
GSS_KRB5_S_G_WRONG_SIZE         /* "La mémoire tampon n'a pas la bonne taille" */
GSS_KRB5_S_G_BAD_USAGE          /* "Le type d'usage d'accréditif est inconnu" */
GSS_KRB5_S_G_UNKNOWN_QOP        /* "Qualité de protection spécifiée inconnue" */
```

5.1.2 Codes spécifiques de Kerberos

```
GSS_KRB5_S_KG_CCACHE_NOMATCH        /* "Le client principal dans les accréditifs ne correspond pas au nom
                                    spécifié" */
GSS_KRB5_S_KG_KEYTAB_NOMATCH        /* "Pas de clé disponible pour le principal de service spécifié" */
GSS_KRB5_S_KG_TGT_MISSING           /* "Pas de ticket d'accord de ticket Kerberos disponible" */
GSS_KRB5_S_KG_NO_SUBKEY             /* "L'authentificateur n'a pas de sous clé" */
GSS_KRB5_S_KG_CONTEXT_ESTABLISHED   /* "Le contexte est déjà pleinement établi" */
GSS_KRB5_S_KG_BAD_SIGN_TYPE         /* "Type de signature inconnu dans le jeton" */
GSS_KRB5_S_KG_BAD_LENGTH            /* "Longueur de champ invalide dans le jeton" */
GSS_KRB5_S_KG_CTX_INCOMPLETE        /* "Tentative d'utiliser un contexte de sécurité incomplet" */
```

5.2 Tailles de mémoire tampon

Toutes les mises en œuvre de cette spécification DOIVENT être capables d'accepter des mémoires tampon d'au moins 16 koctets en entrée à GSS_GetMIC(), GSS_VerifyMIC(), et GSS_Wrap(). Elles DOIVENT aussi être capables d'accepter le jeton de résultat généré par GSS_Wrap() pour une mémoire tampon d'entrée de 16 koctets en entrée à GSS_Unwrap(). Les mises en œuvre DEVRAIENT accepter des mémoires tampon d'entrée de 64 koctets, et PEUVENT accepter des tailles de mémoire tampon d'entrée encore plus grandes.

6. Considérations de rétro compatibilité

Les nouveaux formats de jetons définis dans le présent document ne seront reconnus que par les nouvelles mises en œuvre. Pour traiter cela, les mises en œuvre peuvent toujours utiliser l'algorithme de signature ou sceau explicite de la [RFC1964] quand le type de clé correspond à des "enctypes" non nouveaux. Autrement, on peut réessayer d'envoyer le message avec l'algorithme de signature ou sceau explicitement défini comme dans la [RFC1964]. Cependant, cela va exiger l'utilisation d'un mécanisme comme celui de la [RFC2478] pour négocier la méthode en toute sécurité, ou l'utilisation d'un mécanisme hors bande pour choisir le mécanisme approprié. Pour cette raison, il est RECOMMANDÉ que les nouveaux formats de jetons définis dans ce document soient utilisés seulement si les deux homologues sont connus pour prendre en charge le nouveau mécanisme durant la négociation de contexte à cause de, par exemple, l'utilisation de "nouveaux" enctypes.

GSS_Unwrap() ou GSS_VerifyMIC() peut traiter un jeton de message comme suit : il peut regarder le premier octet de l'en-tête de jeton, et si il est 0x60, le jeton doit alors porter le tramage générique GSS-API pseudo ASN.1. Autrement, les deux premiers octets du jeton contiennent le TOK_ID qui identifie de façon univoque le format de message du jeton.

7. Considérations pour la sécurité

Les liens de canal sont validés par l'accepteur. L'accepteur peut ignorer la restriction de lien de canal fournie par l'initiateur et portée dans la somme de contrôle de l'authentificateur, si (1) les liens de canal ne sont pas utilisés par GSS_Accept_sec_context [RFC2743], et (2) l'accepteur ne prouve pas à l'initiateur qu'il a les mêmes liens de canal que l'initiateur (même si le client a demandé l'authentification mutuelle). Cette limitation devrait être prise en compte par les concepteurs d'applications qui voudraient utiliser les liens de canal, pour limiter l'utilisation des contextes GSS-API aux nœuds avec des adresses réseau spécifiques, pour authentifier d'autres canal sûrs établis utilisant Kerberos Version 5, ou pour tout autre objet.

Les types de clé de session sont choisis par le KDC. Avec le mécanisme actuel, aucune négociation des types d'algorithmes n'intervient, de sorte que les mises en œuvre côté serveur (accepteur) ne peuvent pas demander que les clients n'utilisent pas des types d'algorithmes non compris par le serveur. Cependant, les administrateurs peuvent contrôler quels enctypes peuvent être utilisés pour les clés de session pour ce mécanisme en contrôlant l'ensemble de enctypes de clé de session de ticket que le KDC veut utiliser dans les tickets pour un certain principal accepteur. Donc, le KDC pourrait recevoir la tâche de limiter les clés de session pour un certain service aux types réellement pris en charge par le logiciel Kerberos et GSSAPI sur le serveur. Cela a pour inconvénient que dans les cas où un nom de principal de service est utilisé à la fois pour des communications fondées sur GSSAPI et des communication non fondées sur GSSAPI (en particulier la clé de service "hôte") si la mise en œuvre GSSAPI ne comprend pas (par exemple) AES [RFC3962], mais si la mise en œuvre Kerberos le comprend. Cela signifie que les clés de session AES ne peuvent pas être produites pour ce principal de service, qui conserve la protection de services non GSSAPI plus faible que nécessaire. Les administrateurs de KDC qui désirent limiter les types de clé de session pour prendre en charge l'interopérabilité avec de telles mises en œuvre GSSAPI devraient soupeser avec soin la réduction de protection offerte par de tels mécanismes par rapport aux avantages de l'interopérabilité.

8 Remerciements

Ken Raeburn et Nicolas Williams ont corrigé beaucoup de nos erreurs dans l'utilisation des profils génériques et ont joué un rôle essentiel dans la création de ce document.

Le texte des considérations pour la sécurité a été écrit par Nicolas Williams et Ken Raeburn.

Sam Hartman et Ken Raeburn ont suggéré l'idée de "l'en-queue flottant", à savoir le codage du champ RRC.

Sam Hartman et Nicolas Williams ont recommandé de remplacer notre fonction de déduction de clé antérieure par des clés directionnelles avec des numéros d'usage de clé différents pour chaque direction ainsi que de conserver le bit directionnel pour une compatibilité maximale.

Paul Leach a fourni de nombreuses suggestions et commentaires.

Scott Field, Richard Ward, Dan Simon, Kevin Damour, et Simon Josefsson ont aussi fourni de précieux apports à ce document.

Jeffrey Hutzelman a fourni des commentaires et des précisions pour le texte relatif aux liens de canal.

Jeffrey Hutzelman et Russ Housley ont suggéré de nombreux changements rédactionnels.

Luke Howard a fourni les mise en œuvre de ce document pou la base de code Heimdal, et a aidé aux essais d'interopérabilité avec la base de code Microsoft, avec Love Hornquist Astrand. Ces expériences ont formé la base de ce document.

Martin Rex a fourni des suggestions de recommandations d'allocation de TOK_ID, et don l'étiquetage des jetons dans ce document est sans ambiguïté si le jeton est enveloppé avec le pseudo en-tête ASN.1.

John Linn a écrit la spécification du mécanisme Kerberos Version 5 original [RFC1964], dont du texte a été conservé.

9. Références

9.1 Références normatives

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

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

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

[RFC2744] J. Wray, "API de service générique de sécurité, version 2 : liaisons C", janvier 2000. *(P.S.)*

[RFC3961] K. Raeburn, "Spécifications de chiffrement et de somme de contrôle pour Kerberos 5", février 2005. *(MàJ par* 8429

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

9.2 Références pour information

[RFC2478] E. Baize, D. Pinkas, "Mécanisme de négociation GSS-API simple et protégé", décembre 1998. (*Obs., voir* RFC4178) *PS*

[RFC3962] K. Raeburn, "Chiffrement de la norme de chiffrement évolué (AES) pour Kerberos 5", février 2005.

[SSPI] Leach, P., "Security Service Provider Interface", Microsoft Developer Network (MSDN), avril 2003.

Adresse des auteurs

Larry Zhu
One Microsoft Way
Redmond, WA 98052 - USA
mél : LZhu@microsoft.com

Karthik Jaganathan
One Microsoft Way
Redmond, WA 98052 - USA
mél : karthikj@microsoft.com

Sam Hartman
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139 - USA
mél : hartmans-ietf@mit.edu

Déclaration complète de droits de reproduction

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