Groupe de travail Réseau
Request for Comments : 5403
Met à jour la RFC 2203
Catégorie : Sur la voie de la normalisation

M. Eisler, NetApp
février 2009

Traduction Claude Brière de L'Isle

RPCSEC_GSS Version 2

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet sur la voie de la normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document décrit la version 2 du protocole RPCSEC_GSS. La version 2 est la même que la version 1 (spécifiée dans la RFC 2203) excepté que la prise en charge des liens de canaux a été ajoutée. RPCSEC_GSS permet à des protocoles d'appel de procédure à distance (RPC, *Remote Procedure Call*) d'accéder aux interfaces de programmation de service de sécurité générique (GSS-API, *Generic Security Services Application Programming Interface*).

Table des matières

1. Introduction et motifs

Le présent document décrit la version 2 de RPCSEC_GSS (RPCSEC_GSSv2). RPCSEC_GSSv2 est le même que RPCSEC_GSSv1 [RFC2203] excepté la prise en charge des liens de canal [RFC5056] qui y a été ajoutée. La principale motivation des liens de canal est de tirer parti en toute sécurité du chiffrement assisté par le matériel qui peut exister à des niveaux inférieurs de la pile de protocoles de réseautage, comme à la couche IP sous la forme de IPsec (voir la [RFC5660] et [Williams] pour des informations sur les liens de canal IPsec). Le second motif est que même si les niveaux inférieurs ne sont pas plus efficaces au chiffrement que la couche RPCSEC_GSS, si le chiffrement se produit au niveau inférieur, il peut

être redondant au niveau RPCSEC_GSS.

RPCSEC_GSSv2 et RPCSEC_GSSv1 sont des protocoles qui échangent des jetons émis par le cadre des services génériques de sécurité (GSS, *Generic Security Services*) qui est défini dans la [RFC2743], et diffèrent seulement dans la prise en charge des liens de canal GSS dans RPCSEC_GSSv2. GSS lui-même prend en charge les liens de canal, et en théorie RPCSEC_GSSv2 pourrait utiliser des liens de canal natifs GSS pour réaliser les effets décrits dans cette section. Cependant, comme le déclare le paragraphe 1.1.6 de la [RFC2743], toutes les mises en œuvre de tous les mécanismes GSS ne prennent pas en charge les liens de canal. C'est une justification suffisante pour l'approche prise dans le présent document : modifier le protocole RPCSEC_GSS pour qu'il prenne en charge les liens de canal indépendamment des capacités du mécanisme GSS utilisé.

Une fois qu'une cible et un initiateur RPCSEC_GSS se sont mutuellement assurés qu'ils utilisent chacun le même canal sûr de bout en bout, les frais généraux du calcul des codes d'intégrité de message (MIC, *Message Integrity Code*) pour l'authentification et la protection de l'intégrité des demandes et réponses RPC peuvent être éliminés parce que le canal effectue la même fonction. De même, si le canal assure aussi la confidentialité, les frais généraux de la protection de la confidentialité de RPCSEC_GSS peuvent aussi être éliminés.

La description en représentation de données externes (XDR, *External Data Representation*) [RFC4506] est fournie dans ce document d'une façon qui rend simple au lecteur de l'extraire sous une forme prête à être compilée. Le lecteur peut entrer ce document dans le script suivant pour produire la description XDR lisible par la machine de RPCSEC_GSSv2 :

<DÉBUT DU CODE>

```
#!/bin/sh
grep "^  *///" | sed 's?^  *///??'
```

<FIN DU CODE>

C'est-à-dire que si le script ci-dessus est mémorisé dans un fichier appelé "extract.sh", et si ce document est dans un fichier appelé "spec.txt", le lecteur peut alors faire :

<DÉBUT DU CODE>

```
sh extract.sh < spec.txt > rpcsec_gss_v2.x
```

<FIN DU CODE>

L'effet du script est de supprimer l'espace en tête de chaque ligne de la spécification, plus une séquence sentinelle de "///".

1.1 Langage des exigences

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].

2. Explication des liens de canaux

Si un canal entre deux parties est sûr, il doit y avoir des informations partagées entre les deux parties. Ces informations pourraient être secrètes ou non. Les exigences de secret dépendent des spécificités du canal.

Par exemple, les informations partagées pourraient être l'enchaînement de la clé publique de la source et de la destination du canal (où chaque clé publique a une clé privée correspondante). Supposons que le canal n'est pas de bout en bout, c'est-à-dire, un interposé (MITM, *man-in-the-middle*) existe, et il y a deux canaux, un de l'initiateur au MITM, et un du MITM à la cible. Le MITM ne peut pas simplement forcer chaque canal à utiliser les mêmes clés publiques, parce qu'une clé publique dérive d'une clé privée, et le système de gestion de clé pour chaque nœud va sûrement allouer des clés privées uniques ou aléatoires. Au plus, le MITM peut forcer une extrémité de chaque canal à utiliser la même clé publique. Le MIC des clés publiques provenant de l'initiateur ne va pas être vérifié par la cible, parce qu'au moins une des clés publiques va être différente. De même, le MIC des clés publiques provenant de la cible ne va pas être vérifié par l'initiateur parce que au

moins une des clés publiques va être différente.

Un protocole de couche supérieure qui utilise le canal sûr peut en toute sécurité exploiter le canal au bénéfice mutuel des parties du niveau supérieur si chaque partie de niveau supérieur peut prouver :
- o que chacune connaît les informations partagées du canal,
- o que la preuve de la connaissance des informations partagées est en fait portée par chacune des parties de niveau supérieur, et pas par d'autres entités.

RPCSEC_GSSv2 ajoute simplement un aller-retour facultatif où l'initiateur calcule un MIC GSS sur les informations partagées du lien de canal, et envoie le MIC à la cible. La cible vérifie le MIC, et à son tour envoie son propre MIC des informations partagées à l'initiateur qui vérifie alors le MIC de la cible. Cela réalise trois choses. D'abord que l'initiateur et la cible sont mutuellement authentifiés. Ensuite, que l'initiateur et la cible prouvent qu'ils connaissent les informations partagées du canal, et donc utilisent le même canal. Enfin que la première et la seconde chose sont faites simultanément.

3. Protocole RPCSEC_GSSv2

Le protocole RPCSEC_GSSv2 va maintenant être expliqué. Le protocole entier n'est pas présenté. On montre à la place les différences entre RPCSEC_GSSv2 et RPCSEC_GSSv1.

3.1 Compatibilité avec RPCSEC_GSSv1

La fonction de RPCSEC_GSSv1 est entièrement prise en charge par RPCSEC_GSSv2.

3.2 Nouveau numéro de version

<DÉBUT DU CODE>

```
/// /*
///  * Copyright (c) 2009 IETF Trust et les personnes identifiées comme auteurs du document. Tous droits réservés.
///  *
///  * Les auteurs du document sont identifiés dans les [RFC2203] et [RFC5403].
///  *
///  * La redistribution et l'utilisation en forme source et binaire, avec ou sans modification, sont permises pourvu que
///  * les conditions suivantes soient satisfaites :
///  * o la redistribution du code source doit conserver la notice de droits de reproduction ci-dessus, cette liste de conditions,
///  *    et le déclinatoire de responsabilité suivant ;
///  * o la redistribution en forme binaire doit reproduire la notice de droits de reproduction, cette liste de conditions
///  *    et le déclinatoire de responsabilité suivant dans la documentation et/ou autre matériaux fournis avec la distribution ;
///  * o ni le nom de Internet Society, IETF ou IETF Trust, ni les noms des contributeurs spécifiques, ne peuvent être
///  *    utilisés pour endosser ou promouvoir des produits dérivés de ce logiciel sans permission écrite préalable.
///  *
///  *   Ce logiciel est fourni par les détenteurs des droits de reproduction et les contributeurs "tel qu'il est" et toutes
///  *   garanties expresses ou implicites, incluant, mais non limitées, les garanties implicites de commercialisabilité et de
///  *   convenance pour un objet particulier sont déclinées, et en aucun cas le détenteur des droits de reproduction ou les
///  *  contributeurs ne sont responsables de dommages directs, indirects, incidents, particuliers, exemplaires, ou
///  *  conséquents (incluant, mais non limités à, la fourniture de biens ou services de substitution ; perte d'usage, données,
///  *   ou profits ; ou d'interruption d'affaires) cependant causés et de toute théorie de responsabilité, par contrat, par
///  *   responsabilité stricte ou par tort (incluant la négligence ou autre) survenant d'une manière quelconque de l'utilisation
///  *  de ce logiciel, même si il est averti de la possibilité d'un tel dommage.
///  */
/// /*
///  * Ce code a été dérivé de la [RFC2203]. Prière de reproduire cette note si possible.
///  */
///
///  enum rpc_gss_service_t {
///                                                    /* Note : la valeur pour 0 est réservée. */
///    rpc_gss_svc_none        = 1,
///    rpc_gss_svc_integrity    = 2,
```

```
///    rpc_gss_svc_privacy      = 3,
///    rpc_gss_svc_channel_prot = 4                    /* nouveau */
/// };
///
///  enum rpc_gss_proc_t {
///    RPCSEC_GSS_DATA                = 0,
///    RPCSEC_GSS_INIT                = 1,
///    RPCSEC_GSS_CONTINUE_INIT = 2,
///    RPCSEC_GSS_DESTROY          = 3,
///    RPCSEC_GSS_BIND_CHANNEL  = 4              /* nouveau */
/// };
///
/// struct rpc_gss_cred_vers_1_t {
///    rpc_gss_proc_t    gss_proc;                  /* procédure de contrôle */
///    unsigned int      seq_num;                   /* numéro de séquence */
///    rpc_gss_service_t service;                   /* service utilisé */
///    opaque            handle<>;                  /* bride de contexte */
/// };
///
/// const RPCSEC_GSS_VERS_1 = 1;
/// const RPCSEC_GSS_VERS_2 = 2;                   /* nouveau */
///
/// union rpc_gss_cred_t switch (unsigned int rgc_version) {
///    case RPCSEC_GSS_VERS_1:
///    case RPCSEC_GSS_VERS_2:                     /* nouveau */
///      rpc_gss_cred_vers_1_t rgc_cred_v1;
/// };
///
```

<FIN DU CODE>

Figure 1

Comme il est apparent ci-dessus, l'accréditif RPCSEC_GSSv2 a le même format que l'accréditif RPCSEC_GSSv1 (bien que corrigé pour que la définition soit en langage de description XDR légal qui est aussi compatible avec la [RFC1831] ; donc, le champ "version", un mot clé dans la RFC 1831, est remplacé par "rgc_version"). Régler le champ rgc_version à 2 indique que l'initiateur et la cible prennent en charge les liens de canal.

3.3 Nouvelle procédure - RPCSEC_GSS_BIND_CHANNEL

<DÉBUT DU CODE>

```
/// struct rgss2_bind_chan_MIC_in_args {
///    opaque         rbcmia_bind_chan_hash<>;
/// };
///
/// typedef opaque    rgss2_chan_pref<>;
/// typedef opaque    rgss2_oid<>;
///
/// struct rgss2_bind_chan_verf_args {
///    rgss2_chan_pref rbcva_chan_bind_prefix;
///    rgss2_oid       rbcva_chan_bind_oid_hash;
///    opaque          rbcva_chan_mic<>;
/// };
///
```

<FIN DU CODE>

Figure 2

Une fois qu'une bride RPCSEC_GSSv2 a été établie sur un canal sûr, l'initiateur PEUT produire RPCSEC_GSS_BIND_CHANNEL (Figure 1). Les cibles DOIVENT prendre en charge RPCSEC_GSS_BIND_CHANNEL. Comme pour les demandes RPCSEC_GSS_INIT et RPCSEC_GSS_CONTINUE_INIT, la procédure NULL RPC DOIT être utilisée. À la différence de ces deux demandes, les arguments de la procédure NULL ne sont pas surchargés, parce que le vérificateur est de taille suffisante pour les besoins de RPCSEC_GSS_BIND_CHANNEL. Le champ gss_proc est réglé à RPCSEC_GSS_BIND_CHANNEL. Le champ seq_num est réglé comme si gss_proc était réglé à RPCSEC_GSS_DATA. Le champ Service est réglé à rpc_gss_svc_none. Le champ "handle" est réglé à une bride RPCSEC_GSS comme retourné par RPCSEC_GSS_INIT ou RPCSEC_GSS_CONTINUE_INIT.

La demande RPCSEC_GSS_BIND_CHANNEL est similaire à la demande RPCSEC_GSS_DATA en ce que leurs vérificateurs contiennent tous deux des MIC. Comme décrit au paragraphe 5.3.1 de la [RFC2203], quand gss_proc est RPCSEC_GSS_DATA, le vérificateur d'une demande RPC est réglé au résultat de GSS_GetMIC() sur l'en-tête RPC. Quand gss_proc est RPCSEC_GSS_BIND_CHANNEL le vérificateur d'une demande RPC est réglé au codage XDR sur une valeur de type de données rgss2_bind_chan_verf_args, qui inclut un MIC comme décrit ci-dessous. Le type de données rgss2_bind_chan_verf_args consiste en trois champs :

- rbcva_chan_bind_prefix. C'est le préfixe de lien de canal comme décrit dans la [RFC5056] jusqu'à, exclu, les deux-points (ASCII 0x3A) qui séparent le préfixe du suffixe.

- rbcva_chan_bind_hash_oid. C'est l'identifiant d'objet (OID) de l'algorithme de hachage utilisé pour calculer rbcmia_bind_chan_hash. Ce champ contient un OID codé en ASN.1 comme utilisé par GSS-API dans l'argument mech_type à GSS_Init_sec_context ([RFC2743]). Voir la [RFC4055] pour les OID des algorithmes de hachage unidirectionnel SHA.

- rbcva_chan_mic. C'est le résultat de GSS_GetMIC() sur l'enchaînement de l'en-tête RPC codé en XDR ("jusqu'à et y compris l'accréditif" conformément à la [RFC2203]) et le codage XDR d'une instance de données de type rgss2_bind_chan_MIC_in_args. Le type de données rgss2_bind_chan_MIC_in_args consiste en un champ, rbcmia_bind_chan_hash, qui est un hachage des liens de canal comme défini dans la [RFC5056]. Les liens de canal sont un "codage canonique de chaîne d'octets des liens de canal", commençant "par le préfixe de liens de canal suivi par un caractère deux-points (ASCII 0x3A)". La raison d'un hachage des liens de canal plutôt que les liens de canal réels utilisé pour calculer rbcva_chan_mic est que certains liens de canal, comme ceux composés de clés publiques, peuvent être relativement grands, et donc faire peser une charge d'espace élevée sur les mises en œuvre. Les hachages unidirectionnels consomment moins d'espace.

  <DÉBUT DU CODE>

```
/// enum rgss2_bind_chan_status {
///   RGSS2_BIND_CHAN_OK                = 0,
///   RGSS2_BIND_CHAN_PREF_NOTSUPP  = 1,
///   RGSS2_BIND_CHAN_HASH_NOTSUPP = 2
/// };
///
/// union rgss2_bind_chan_res switch
///    (rgss2_bind_chan_status rbcr_stat) {
///
///   case RGSS2_BIND_CHAN_OK:
///     void;
///
///   case RGSS2_BIND_CHAN_PREF_NOTSUPP:
///     rgss2_chan_pref rbcr_pref_list<>;
///
///   case RGSS2_BIND_CHAN_HASH_NOTSUPP:
///     rgss2_oid       rbcr_oid_list<>;
/// };
///
/// struct rgss2_bind_chan_MIC_in_res {
///   unsigned int       rbcmr_seq_num;
///   opaque             rbcmr_bind_chan_hash<>;
///   rgss2_bind_chan_res rbcmr_res;
/// };
```

```
///

/// struct rgss2_bind_chan_verf_res {
///   rgss2_bind_chan_res rbcvr_res;
///   opaque              rbcvr_mic<>;
/// };
///
```

<FIN DU CODE>

Figure 3

La réponse RPCSEC_GSS_BIND_CHANNEL est similaire à la réponse RPCSEC_GSS_DATA en ce que les vérificateurs des deux contiennent des MIC. Quand gss_proc est RPCSEC_GSS_DATA, le vérificateur d'une réponse RPC est réglé au résultat de GSS_GetMIC() sur le seq_num de l'accréditif de la demande correspondante (comme décrit au paragraphe 5.3.3.2 de la [RFC2203]). Quand gss_proc est RPCSEC_GSS_BIND_CHANNEL, le vérificateur d'une réponse RPC est réglé au codage XDR d'une instance de type de données rgss2_bind_chan_verf_res, qui inclut un MIC comme décrit ci-dessous. Le type de données rgss2_bind_chan_verf_res consiste en deux champs :

o rbcvr_res. Le type de données de ce champ est rgss2_bind_chan_res. Le type de données rgss2_bind_chan_res est une union commutée consistant en trois cas commutés sur l'état contenu dans le champ rbcr_stat.

  * RGSS2_BIND_CHAN_OK. Si cet état est retourné, la cible a accepté les liens de canal, et a réussi à vérifier rbcva_chan_mic dans la demande. Aucun résultat supplémentaire ne sera dans rbcvr_res.

  * RGSS2_BIND_CHAN_PREF_NOTSUPP. Si cet état est retourné, la cible ne prenait pas en charge le préfixe dans le champ rbcva_chan_bind_prefix des arguments, et donc la demande RPCSEC_GSS_BIND_CHANNEL a été rejetée. La cible a retourné une liste de préfixes qu'elle prend en charge dans le champ rbcr_pref_list. Noter qu'un canal peut avoir plusieurs liens de canal avec chacun des préfixes différents. L'initiateur est libre de prendre son préfixe préféré. Si la cible ne prend pas en charge le préfixe, l'état RGSS2_BIND_CHAN_PREF_NOTSUPP va être retourné, et l'initiateur peut choisir le prochain préfixe préféré parmi les préfixes que la cible prend en charge.

  * RGSS2_BIND_CHAN_HASH_NOTSUPP. Si cet état est retourné, la cible ne prenait pas en charge l'algorithme de hachage identifié dans le champ rbcva_chan_bind_hash_oid des arguments, et donc la demande RPCSEC_GSS_BIND_CHANNEL a été rejetée. La cible a retournée une liste des OID d'algorithmes de hachage qu'elle prend en charge dans le champ rbcr_oid_list. Le dispositif rbcr_oid_list DOIT avoir un ou plusieurs éléments.

o bcvr_mic. La valeur de ce champ est égale au résultat de GSS_GetMIC() sur le codage XDR d'une instance de type de données rgss2_bind_chan_MIC_in_res. Le type de données rgss2_bind_chan_MIC_in_res consiste en trois champs :

  * rbcmr_seq_num. La valeur de ce champ est égale au champ seq_num dans l'accréditif RPCSEC_GSS (type de données rpc_gss_cred_vers_1_t).

  * rbcmr_bind_chan_hash. C'est le résultat du hachage unidirectionnel des liens de canal (incluant le préfixe). Si rbcr_stat n'est pas RGSS2_BIND_CHAN_HASH_NOTSUPP, alors l'algorithme de hachage qui est utilisé pour calculer rbcmr_bind_chan_hash est celui identifié par le champ rbcva_chan_bind_oid_hash dans les arguments à RPCSEC_GSS_BIND_CHANNEL. Si rbcr_stat est RGSS2_BIND_CHAN_HASH_NOTSUPP, alors l'algorithme de hachage utilisé pour calculer rbcmr_bind_chan_hash est celui identifié par rbcr_oid_list[0] dans le résultat.

  * rbcmr_res. La valeur de ce champ est égale à la valeur du champ rbcvr_res.

3.4 Nouveau service de sécurité - rpc_gss_svc_channel_prot

Les cibles RPCSEC_GSSv2 DOIVENT prendre en charge rpc_gss_svc_channel_prot.

Le service rpc_gss_svc_channel_prot (Figure 1) est valide seulement si RPCSEC_GSSv2 est utilisé, une procédure RPCSEC_GSS_BIND_CHANNEL a été exécutée avec succès, et le canal sûr existe encore. Quand rpc_gss_svc_channel_prot est utilisé, les demandes et réponses RPC sont similaires à celles de rpc_gss_svc_none sauf que les vérificateurs sur les demandes et réponses ont toujours la version réglée à AUTH_NONE, et le contenu est de longueur

zéro.

Noter que même si des vérificateurs NULS sont utilisés quand rpc_gss_svc_channel_prot est utilisé, des accréditifs RPCSEC_GSS non NULS sont utilisés. Afin d'identifier le principal qui envoie la demande, le même accréditif est utilisé comme avant, sauf que le champ service est réglé à rpc_gss_svc_channel_prot.

4. Négociation de version

Un initiateur qui prend en charge la version 2 de RPCSEC_GSS produit simplement une demande RPCSEC_GSS avec le champ rgc_version réglé à RPCSEC_GSS_VERS_2. Si la cible ne reconnaît pas RPCSEC_GSS_VERS_2, elle va retourner une erreur RPC selon le paragraphe 5.1 de la [RFC2203].

L'initiateur NE DOIT PAS tenter d'utiliser une bride RPCSEC_GSS retournée par la version 2 d'une cible avec version 1 de la même cible. L'initiateur NE DOIT PAS tenter d'utiliser une bride RPCSEC_GSS retournée par la version 1 d'une cible avec version 2 de la même cible.

5. Liens de canaux GSS natifs

Pour assurer l'interopérabilité, les mises en œuvre de RPCSEC_GSSv2 NE DEVRAIENT PAS transférer de jetons entre l'initiateur et la cible qui utilisent des liens de canal GSS natifs (comme défini au paragraphe 1.1.6 de la [RFC2743]).

6. Recommandation opérationnelle pour le déploiement

RPCSEC_GSSv2 est un sur-ensemble de RPCSEC_GSSv1, et donc peut être utilisé dans toutes les situations où RPCSEC_GSSv1 est utilisé. RPCSEC_GSSv2 devrait être utilisé quand la nouvelle fonctionnalité, liens de canal, est désirée ou nécessaire.

7. Notes de mise en œuvre

Une fois qu'une procédure RPCSEC_GSS_BIND_CHANNEL réussie a été effectuée sur une bride de contexte RPCSEC_GSSv2, la mise en œuvre d'initiateur peut transposer les demandes d'application pour rpc_gss_svc_none et rpc_gss_svc_integrity en accréditifs rpc_gss_svc_channel_prot. Et si le canal sûr a la confidentialité activée, les demandes pour rpc_gss_svc_privacy peuvent aussi être transposées en rpc_gss_svc_channel_prot.

8. Remerciements

Nicolas Williams a eu l'idée d'étendre RPCSEC_GSS pour prendre en charge les liens de canaux. Alex Burlyga, Lars Eggert, Pasi Eronen, et Dan Romascanu on relu le document et donné de précieux retour pour améliorer sa lisibilité.

9. Considérations sur la sécurité

Les considérations de base sur la sécurité consistent en :
- o Toutes les considérations sur la sécurité provenant de la [RFC2203].
- o Toutes les considérations sur la sécurité provenant de la [RFC5056].
- o Toutes les considérations sur la sécurité provenant du canal sûr actuel utilisé.

Bien que les demandes RPCSEC_GSS_DATA qui utilisent la protection rpc_gss_svc_channel_prot n'impliquent pas la construction de plus de jetons GSS, la cible DEVRAIT arrêter de permettre les demandes RPCSEC_GSS_DATA avec la protection rpc_gss_svc_channel_prot une fois que le contexte GSS expire.

Avec l'utilisation des liens de canal, il devient extrêmement critique que le code d'intégrité de message (MIC, *message integrity code*) du mécanisme GSS qu'utilise RPCSEC_GSS soit difficile à falsifier. Alors que cette exigence est vraie pour RPCSEC_GSSv1, et bien sûr tout protocole qui utilise les MIC GSS, la différence est sérieuse pour RPCSEC_GSSv1, car la falsification d'un seul MIC permet à l'attaquant de réussir à injecter une demande boguée. Tandis qu'avec RPCSEC_GSSv2 combiné aux liens de canal, en falsifiant un seul MIC, l'attaquant va réussir à injecter des demandes boguées tant que le canal existe. Un exemple l'illustre. Supposons qu'on ait un initiateur RPCSEC_GSSv1, un attaquant interposé (MITM), une cible RPCSEC_GSSv1, et une cible RPCSEC_GSSv2. L'attaque est comme suit :

- Le MITM intercepte le message RPCSEC_GSS_INIT RPCSEC_GSSv1 de l'initiateur et change le numéro de version de 1 à 2 avant de le transmettre à la cible RPCSEC_GSSv2, et change le numéro de version de la réponse de 2 à 1 avant de la transmettre à l'initiateur RPCSEC_GSSv1. Ni le client ni le serveur ne le remarquent.

- Une fois que la bride RPCSEC_GSS est dans un état établi, l'initiateur envoie sa première demande RPCSEC_GSS_DATA. Le MITM construit une demande RPCSEC_GSS_BIND_CHANNEL, en utilisant le code d'intégrité de message (MIC) de la demande RPCSEC_GSS_DATA. Il est probable que la cible RPCSEC_GSSv2 va rejeter la demande. Le MITM va réitérer chaque fois que l'initiateur envoie une autre demande RPCSEC_GSS_DATA. Avec assez d'itérations, la probabilité qu'un MIC d'une RPCSEC_GSS_DATA soit vérifié avec succès dans le RPCSEC_GSS_BIND_CHANNEL falsifié augmente. Une fois que le MITM a réussi, il peut envoyer des demandes RPCSEC_GSS_DATA avec un service de sécurité de rpc_gss_svc_channel_prot, qui n'a pas de MIC dans le vérificateur de la demande RPC.

La mise en œuvre de RPCSEC_GSSv2 peut utiliser au moins deux méthodes pour déjouer ces attaques :

- La cible DEVRAIT exiger un MIC plus fort quand elle envoie une demande RPCSEC_GSS_BIND_CHANNEL au lieu d'une demande RPCSEC_GSS_DATA -- par exemple, si des HMAC sont utilisés pour les MIC, exiger le HMAC le plus grand possible (en terme de nombre de bits) que le mécanisme GSS accepte. Si des HMAC sont utilisés, et si la cible s'attend à ce que N demandes RPCSEC_GSS_DATA soient envoyées sur le contexte avant qu'il expire, alors la cible DEVRAIT exiger un HMAC pour RPCSEC_GSS_BIND_CHANNEL qui est log base 2 N bits plus long que ce qui est normalement exigé pour les demandes RPCSEC_GSS_DATA. Si un MIC assez long n'est pas disponible, alors la cible pourrait artificiellement limiter le nombre de demandes RPCSEC_GSS_DATA qu'elle va permettre sur le contexte avant de supprimer le contexte.

- Chaque fois qu'une cible RPCSEC_GSSv2 subit un échec de vérification du MIC d'une demande RPCSEC_GSS_BIND_CHANNEL, elle DEVRAIT réduire la durée de vie du contexte GSS sous-jacent, d'une fraction significative, empêchant par là le MITM d'utiliser le contexte établi pour son attaque. Une heuristique possible est que si la cible estime possible à X pour cent que l'échec de vérification du MIC soit causé par une attaque, alors la durée de vie du contexte devrait être réduite de X pour cent. Pour simplifier, une mise en œuvre pourrait régler X à 50 pour cent, de façon que la durée de vie du contexte soit diminuée par deux à chaque échec de vérification d'une demande RPCSEC_GSS_BIND_CHANNEL et soit donc rapidement réduite à zéro sur les demandes suivantes. Par exemple, avec une durée de vie de contexte de 8 heures (ou 28800 secondes) 15 tentatives échouées du MITM vont causer la destruction du contexte.

Une méthode d'atténuation qui a été envisagée était de protéger le numéro de version de RPCSEC_GSS avec des jetons RPCSEC_GSS_INIT et RPCSEC_GSS_CONTINUE_INIT de RPCSEC_GSSv2. Donc, le numéro de version de RPCSEC_GSS va être dans les jetons. Cette méthode ne supprime pas complètement l'attaque, elle déplace juste la conjecture sur le MIC au message RPCSEC_GSS_INIT. De plus, sans changer GSS, ou le mécanisme GSS, il n'y a pas de moyen d'inclure le numéro de version RPCSEC_GSS dans les jetons. C'est pour ces raisons que cette méthode n'a pas été choisie.

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)

[RFC2203] M. Eisler, A. Chiu, L. Ling, "Spécification du protocole RPCSEC_GSS", septembre 1997. *(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)

[RFC4055] J. Schaad et autres, "Algorithmes et identifiants supplémentaires pour la cryptographie RSA à utiliser dans le profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", juin 2005.

[RFC4506] M. Eisler, éd., "XDR : norme de représentation des données externes", mai 2006. (STD0067)

[RFC5056] N. Williams, "Sur l'utilisation des liaisons de canaux pour sécuriser les canaux", novembre 2007. *(P.S.)*

10.2 Références pour information

[RFC1831] R. Srinivasan, "RPC : Spécification de la version 2 du protocole d'appel de procédure à distance", août 1995. *(Expérimentale, Obsolète, voir* RFC5531)

[RFC5660] N. Williams, "Canaux IPsec : verrouillage de connexion**",** octobre 2009. *(P. S.)*

[Williams] Williams, N., "End-Point Channel Bindings for IPsec Using IKEv2 and Public Keys", Travail en cours, avril 2008.

Adresse de l'auteur

Mike Eisler
NetApp
5765 Chase Point Circle
Colorado Springs, CO 80919
US
téléphone : +1-719-599-9026
mél : mike@eisler.com