RFC 6929 : Remote Authentication Dial In User Service (RADIUS) Protocol Extensions

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 mai 2013

Date de publication du RFC : Avril 2013

https://www.bortzmeyer.org/6929.html

RADIUS, un protocole utilisé pour l'authentification (et la configuration) d'accès à l'Internet, est ancien et marche toujours. Lorsque vous accédez à l'Internet depuis chez vous, sans vous en rendre compte, vous avez probablement indirectement utilisé RADIUS. Il marche toujours très bien mais, rançon du succès, il a été tellement étendu qu'un certain nombre de champs du message RADIUS sont désormais pleins : toutes les options possibles seront allouées très bientôt. Ce nouveau RFC décrit un mécanisme d'extension de RADIUS, permettant de continuer l'enregistrement de nouvelles options, au prix de quelques bricolages.

Il y a peu de gens qui manipulent RADIUS (RFC 2865 [1]) explicitement. Ce protocole se trouve en effet caché dans le réseau du FAI, entre la machine qui contrôle l'accès (le NAS) et un serveur RADIUS qui contient les données sur les utilisateurs. RADIUS permet de ne pas avoir à copier les informations d'authentification, ou les paramètres d'un utilisateur donné, directement dans tous les NAS, mais de les centraliser dans un petit nombre de serveurs RADIUS.

La requête, et surtout la réponse RADIUS peuvent contenir un certain nombre d'**attributs** (RFC 2865, section 5) comme « nom de l'utilisateur », « mot de passe », « adresse IP à attribuer », etc. Le champ « "*Attribute*" » est en TLV et le type (`User-Name` = 1, `User-Password` = 2, `Framed-IP-Address` = 8, etc) est encodé sur un seul octet. La liste (un registre IANA <https://www.iana.org/assignments/radius-types/radius-types.xml#radius-types-1>) est aujourd'hui bien remplie et devrait l'être complètement en 2014 ou 2015. C'est le principal problème de RADIUS aujourd'hui (section 1 de notre RFC). Mais il y en a d'autres comme le fait que la longueur d'un attribut soit codé sur un octet seulement (donc 253 octets maximum pour la valeur d'un attribut, puisqu'il faut retirer le type et la longueur).

Notre RFC fait donc les changements suivants :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2865.txt

— Définition du format "*Extended Type*" : certains types d'attributs ont désormais ce format, qui inclut un second champ Type d'un octet dans la partie Valeur de l'attribut. Le nouveau type sera donc indiqué par la combinaison du Type et du nouveau champ (donc, deux octets, et le type sera noté N.M où N et M sont les valeurs des deux octets). Ça fait un peu bricolage, mais cela permet l'interopérabilité entre les nouvelles implémentations et les anciennes. Ce format est utilisé par quatre nouveaux attributs (de 241 à 244) et permet donc environ 1000 types nouveaux. La taille maximale des attributs a donc diminué d'un octet mais le deuxième changement, ci-dessous, résoud cela.
— Définition du format "*Long Extended Type*" qui ajoute un octet (après le type) dans le champ Valeur, pour pouvoir ajouter des nouvelles fonctions au protocole. Deux attributs (245 et 246) utiliseront ce format, soit environ 500 possibilités.
— En utilisant ce format, définition d'un mécanisme permettant de dire que la valeur est stockée sur plusieurs attributs consécutifs. Cette « fragmentation » permettra d'utiliser des attributs de plus de 253 octets. Elle est indiquée par un bit (nommé M pour "*More*", s'il est à Un, cela indique que d'autres attributs suivent) dans le nouvel octet réservé.
— Les types de données disponibles dans le champ Valeur étaient au nombre de cinq (les classiques, nombre entier, adresse IP, temps, texte et suite de bits). Notre RFC 6929 en ajoute deux, TLV (qui va donc permettre de structurer les données puisqu'un TLV pourra en contenir d'autres, et récursivement) et entier de 64 bits (certaines valeurs, par exemple le nombre d'octets transmis, dans les messages de comptabilité, ne tenaient plus toujours en 32 bits).

Voilà, c'est l'essentiel du RFC. Ceux qui étendent le protocole RADIUS vont sans doute migrer vers les nouveaux types, dès aujourd'hui, ou bien lorsque les anciens seront épuisés.

Des exemples d'encodage figurent en section 9. Par exemple, un attribut utilisant un type étendu (ici, 241.1) et ayant une valeur de type texte (la chaîne "bob") se représentera `f1 06 01 62 6f 62` (`f1`, le type, 241, `06` la longueur totale, `01` le sous-type du type étendu, 1 (à la place du premier octet de la valeur), `62 6f 62` la vraie valeur, la chaîne de caractères). Plus compliqué, un attribut 241.2 contenant un TLV dont le type est 1, la longueur 4 et la valeur `23 45`, sera représenté `f1 07 02 01 04 23 45`.

Avec les étendus longs ("*Long Extended Type*"), un attribut 245.1 contenant le même "bob" sera `f5 07 01 00 62 6f 62` (`f5` est le type, la longueur est de sept octets, `01` est la suite du type, l'octet supplémentaire est à 0 (donc le bit M aussi : pas d'autres attributs à attendre) et la valeur est la chaîne "bob" (`62 6f 62` en ASCII). Si un type 245.4 encode plus de 251 octets (ici, 260 octets de valeur), on aura le premier attribut qui commencera par `f5 ff 04 80` (longueur totale à 256 octets, l'octet supplémentaire vaut 80, indiquant que le bit M est à Un) et un deuxième attribut qui commencera par `f5 13 04 00` (octet supplémenaire à 0 donc bit M à Zéro, ce second attribut est le dernier). La longueur totale est `ff + 13` soit 260 octets pour la valeur.

Si vous voulez tester vous-même ces nouveaux types et leur encodage, l'annexe A du RFC contient un programme en C qui permet exactement cela (copie locale du source (en ligne sur `https://www.bortzmeyer.org/files/radius-attr-generator.c`)). Il s'utilise ainsi :

```
% echo '241.1 "bob"' | ./radius-attr-generator
241.1 "bob" -> f1 06 01 62 6f 62

% echo '243.2 42' | ./radius-attr-generator
243.2 42 -> f3 04 02 42

# Plus drôle, un TLV dans un TLV :
% echo '241.4 { 1 23 45 } { 3 { 1 ab cd } }' | ./radius-attr-generator
241.4 { 1 23 45 } { 3 { 1 ab cd } } -> f1 0d 04 01 04 23 45 03 06 01 04 ab cd
```

Parmi les autres nouveautés du RFC, il y a la formalisation de la convention de nommage informel des attributs : mots séparés par des tirets (par exemple `Framed-MTU`), attributs spécifiques à un vendeur préfixés par le nom du vendeur (par exemple `Starbucks-Latte-Size` pour un vendeur nommé Starbucks).

Le changement apporté au protocole n'est pas trivial, mais soigneusement conçu pour maintenir la compatibilité. La section 5 du RFC note toutefois quelques problèmes potentiels. D'abord, certains vendeurs se sont approprié des numéros de types qui étaient marqués comme réservés (et qui sont désormais utilisés pour les types étendus). Comme le notait le RFC 6158, c'est un comportement anti-social. Maintenant que ces numéros réservés sont utilisés, ces attributs spécifiques au vendeur anti-social ne marcheront plus (bien fait).

Autre problème potentiel, il est fréquent que les messages RADIUS soient relayés par un serveur intermédiaire (RFC 2865, section 2.3), notamment en cas d'itinérance. Idéalement, le relais devrait ignorer les attributs qu'il ne comprennent pas. Si, par contre, il rejette les messages contenant ces attributs nouveaux, un client et un serveur conformes au nouveau format ne pourront pas communiquer à travers un tel relais. Notre RFC rappelle bien qu'un relais devrait être transparent et relayer même ce qu'il ne comprend pas.

Les sections 6.6 et 10.3 fournissent les règles d'enregistrement des nouveaux types. Il est recommandé d'y aller mollo : les nouveaux espaces sont plus grands mais pas infinis. Le registre IANA `<https://www.iana.org/assignments/radius-types/radius-types.xml#radius-types-1>` stocke désormais des attributs des nouveaux types.

Devoir ajuster a posteriori un protocole, qui avait été conçu d'une certaine manière, et qui esr assez rigide, est toujours une opération délicate, spécialement lorsqu'il faut maintenir la compatibilité avec l'existant. Des tas de propositions ont été faites depuis des années, pour étendre RADIUS. La section 7 du RFC documente pourquoi celle-ci a été choisie. D'abord, elle est relativement simple, avec peu de modifications du format. Son principal inconvénient est que le format des "*Long Extended Type*" est assez baroque.

Quant au mécanisme pour agrandir la taille possible des valeurs, il a d'abord été conçu pour ne pas changer les attributs dont la valeur restait petite. D'après une étude de 2010 (citée en section 7.1) portant sur tous les dictionnaires Radius connus `<http://github.com/alandekok/freeradius-server/tree/master/share/>`, les entiers représentent près de la moitié des valeurs d'attributs RADIUS publics. En ajoutant les autres types de taille fixe (adresses IP, par exemple), on arrive à plus de la moitié. Il fallait donc éviter de faire payer ces types peu consommateurs en octets.

Plusieurs des mises en œuvre de Radius gèrent déjà ces extensions, par exemple FreeRADIUS. Il faut dire qu'il n'y a guère le choix, vue l'imminence de l'épuisement.