RFC 7617 : The 'Basic' HTTP Authentication Scheme

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 octobre 2015

Date de publication du RFC : Septembre 2015

https://www.bortzmeyer.org/7617.html

Ce court RFC (re-)définit le mécanisme d'authentification « de base » ("*basic authentication scheme*") de HTTP : un mécanisme trivial, où un nom d'utilisateur et un mot de passe sont transmis en Base64 au serveur HTTP. Ce RFC (et ses copains) annule et remplace l'ancien RFC 2617[1].

Depuis la réécriture générale des RFC sur HTTP <https://www.bortzmeyer.org/http-11-reecrit.html>, le cadre de l'authentification est décrit dans le RFC 7235. Chaque mécanisme spécifique fait ensuite l'objet de son propre RFC. L'authentification de base est le plus ancien et ce RFC 7617 n'apporte donc rien de bien nouveau, il se contente de toiletter une norme existante.

Le principe est simple : le serveur HTTP gère un ou plusieurs « royaumes » ("*realm*"). Dans chaque royaume, il connait une liste d'utilisateurs, identifiés par leur nom, et authentifiés par un mot de passe. Le client, à qui le serveur indique le royaume dont dépend la page Web convoitée, transmet cet identificateur et ce mot de passe, et, si cela correspond à ce qui est dans la base, le serveur lui donne accès. Et lui renvoie une code 403 (accès interdit) autrement.

Voici un exemple de dialogue. Le client a tenté d'accéder à la ressource `/rapport.html` qui est protégée. Le serveur répond par le code 401 (authentification nécessaire, cf. RFC 7235, section 3.1) et un défi : « essaie de t'authentifier dans le royaume "*Corporate*" », dans l'en-tête `WWW-Authenticate:` (RFC 7235, section 4.1).

```
HTTP/1.1 401 Unauthorized
Date: Mon, 04 Feb 2014 16:50:53 GMT
WWW-Authenticate: Basic realm="Corporate"
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2617.txt

Le client doit lui répondre avec un nom et un mot de passe. S'il est un navigateur Web, il demandera probablement interactivement à l'utilisateur

Si le client HTTP est un programme autonome, il cherchera dans sa configuration (par exemple, avec curl, ce sera dans le fichier `/.netrc` ou bien dans l'option `--user` de la ligne de commande). Une fois que le client HTTP aura nom et mot de passe, il les concatène (avec un deux-points comme séparateur) et les encode en Base64 (RFC 4648). Si le nom d'utilisateur est `Aladdin` et le mot de passe `open sesame`, voici un moyen simple, avec le shell Unix, de calculer la chaîne de caractères à envoyer :

```
% echo -n "Aladdin:open sesame" | base64
QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Le client HTTP enverra donc sa requête avec l'en-tête :

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

À noter que l'ancienne norme ne spécifiait pas quel encodage utiliser pour représenter nom et mot de passe avant de les passer à Base64. Certains utilisaient Latin-1, d'autres UTF-8. C'est déplorable mais il est trop tard pour changer cela (l'annexe B.3 explique ce choix), donc notre nouveau RFC ne spécifie pas non plus d'encodage, c'est un arrangement privé avec le serveur HTTP, sauf si un encodage a été spécifié dans le défi, avec le paramètre `charset`. Au passage, signalons que la syntaxe possible pour les noms d'utilisateur est désormais celle du RFC 8265.

Ce paramètre `charset` (qui aurait dû se nommer plutôt `accept-charset` car il indique ce que le serveur attend, pas ce qu'il envoie), n'a qu'une seule valeur autorisée, `UTF-8`. Cette valeur indique que le serveur s'attend à ce qu'identificateur et mot de passe soient normalisés en NFC (règles du RFC 5198), puis encodés en UTF-8.

Prenons un exemple, le nom est `test` et le mot de passe `123[Caractère Unicode non montré ]`[2] (le dernier caractère est U+00A3, qui devient C2 A3 en UTF-8, le terminal utilisé pour l'exemple ci-dessous est en UTF-8, ce qui tombe bien). On encode la chaîne `test:123[Caractère Unicode non montré ]` en Base64 :

```
% echo -n test:123£ | base64
dGVzdDoxMjPCow==
```

Et on a donc la valeur à utiliser dans le champ `Authorization:`.

Et le royaume, on peut le mettre en Unicode ? Eh bien, ce point est obscur (section 3 de notre RFC) et il faudra attendre une nouvelle mise à jour pour l'éclaircir.

La section 4 de notre RFC parle de sécurité. Elle rappelle que le mot de passe est transmis en clair (Base64 ne protège évidemment pas du tout) et qu'il est donc impératif de transporter la session HTTP dans du TLS (RFC 2818). La solution du RFC 7616 n'a pas ce défaut.

2. Car trop difficile à faire afficher par LaTeX

Ce mécanisme « de base » souffre également des faiblesses habituelles des mots de passe. Par exemple, si les utilisateurs peuvent les choisir, ils risquent de prendre des mots faibles (prénom, date de naissance...) ou bien de réutiliser le même mot de passe pour plusieurs services. S'ils ne peuvent pas les choisir, ils auront du mal à les mémoriser et seront donc tentés de les écrire... pas forcément dans un endroit protégé.

Dans tous les cas, que les mots de passe soient choisis par les utilisateurs ou pas, le serveur qui les stocke doit prendre des précautions. Ils ne doivent **jamais** être stockés en clair sur le serveur, car, si le serveur est piraté, le pirate aurait alors accès à tous les mots de passe, donc beaucoup pourront être utilisables avec d'autres serveurs. Les mots de passe doivent être stockés condensés et salés.

Le mode "*Basic*" a été enregistré à l'IANA (section 5 du RFC) dans le registre des mécanismes d'authentification `<https://www.iana.org/assignments/http-authschemes/>`.

Quels sont les changements concrets depuis RFC 2617, section 2, qui normalisait ce mécanisme avant ? L'annexe A les décrit : il y a la compatibilité avec le nouveau cadre d'authentification du RFC 7235, et une meilleure internationalisation avec le nouveau paramètre `charset`.