

# Authentification HTTP

L'**authentification HTTP** ou identification HTTP (spécifiée par [RFC 2617]) permet de s'identifier auprès d'un serveur **HTTP** en lui montrant que l'on connaît le nom d'un utilisateur et son mot de passe, afin d'accéder aux ressources à accès restreint de celui-ci.

## 1 Fonctionnement général

Lorsqu'un client HTTP demande une ressource protégée au serveur, celui-ci répond de différente façon selon la requête :

- soit la requête ne contient pas d'en-tête HTTP d'identification, dans ce cas le serveur répond avec le **code HTTP** 401 (*Unauthorized* : non autorisé) et envoie les en-têtes d'information sur l'identification demandée,
- soit la requête contient les en-têtes HTTP d'identification, dans ce cas, après vérification du nom et du mot de passe, si l'identification échoue, le serveur répond par le code 401 comme dans le cas précédent, sinon il répond de manière normale (code 200 OK).

## 2 Méthodes

Il existe deux méthodes définies par la spécification [RFC 2617] :

- la méthode « Basic »,
- la méthode « Digest ».

### 2.1 Méthode « Basic »

Cette méthode est la plus simple, mais également la moins sécurisée car elle transmet le mot de passe en clair (ou presque). Elle n'est recommandée qu'avec une connexion chiffrée (protocole **HTTPS**).

Le serveur ne recevant pas d'en-tête d'identification correcte envoie ce genre d'en-tête HTTP :

WWW-Authenticate : Basic realm="WallyWorld"

Le serveur indique la méthode requise (Basic), suivie des paramètres. La méthode « Basic » ne requiert que le paramètre « realm » identifiant le domaine de protection.

Le client HTTP peut alors réessayer la requête en spécifiant l'en-tête HTTP « Authorization ». Celui-ci doit contenir la méthode utilisée (Basic) suivi de la représentation en **Base64** du nom de l'utilisateur et du mot de passe séparés par le caractère « : » (deux-points).

Par exemple, pour authentifier l'utilisateur « Aladdin » avec le mot de passe « open sesame », le client envoie :

Authorization : Basic QWxhZGRpbjpvcGVuIHNIc2FtZQ==

Base64("Aladdin:open sesame") = "QWxhZGRpbjpvcGVuIHNIc2FtZQ=="

### 2.2 Méthode « Digest »

Cette méthode ne transmet pas le mot de passe en clair, mais impose de stocker celui-ci (ou son hachage SHA1, qui suffit pour s'identifier et peut donc être considéré comme un mot de passe) en clair. Même si cette méthode est plus sûre que la méthode « Basic », elle reste tout de même sensible aux attaques (interception de communication...), et plus sensible encore à certaines attaques (vol de fichier de mots de passe).

La méthode « Digest » est plus complexe et emploie plus de paramètres.

#### 2.2.1 Demande d'identification

Le serveur peut envoyer une demande d'identification du genre :

WWW-Authenticate : Digest realm="testrealm@host.com", qop="auth, auth-int", nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093", opaque="5ccc069c403ebaf9f0171e9517f40e41"

Les paramètres sont séparés par une virgule et sont les suivants :

**realm** Ce paramètre est affiché à l'utilisateur pour qu'il sache quel nom et mot de passe il peut utiliser. La chaîne doit au moins contenir le nom de la machine et le nom du groupe d'utilisateurs requis.

**domain** (*optionnel*) Contient une liste d'**URI** séparés par le caractère espace, définissant le domaine de protection. Ce paramètre donne la liste des URI pour lesquels l'utilisateur demandé est valable.

**nonce** Chaîne générée par le serveur à chaque réponse 401. Il est conseillé que cette chaîne utilise les caractères **Base64** ou hexadécimaux. Ce paramètre est utilisé dans le calcul de la réponse du client.

**opaque** (*optionnel*) Chaîne générée par le serveur que le client doit retourner telle quelle.

**stale** (*optionnel*) Ce paramètre a 2 valeurs possibles : **true** ou **false**. Il vaut **true** si la demande d'identification précédente a été rejetée seulement à cause de l'utilisation d'une ancienne valeur du paramètre « nonce », **false** sinon. La valeur **true** indique que le client doit retenter la requête en utilisant la nouvelle valeur de **nonce** fournie par le serveur sans redemander un nom et un mot de passe à l'utilisateur.

**algorithm** (*optionnel*) Indique l'algorithme à utiliser pour les fonctions de hashages. Deux valeurs sont définies dans [RFC 2617] : **MD5** ou **MD5-sess**.

**qop** (*optionnel*) (Quality Of Protection) Ce paramètre indique les niveaux de protection supportés : **auth** ou **auth-int**.

### 2.2.2 Identification du client

Le client s'identifiant avec la méthode « Digest » utilise deux fonctions pour calculer certains paramètres :

$H(data) = MD5(data)$

La fonction H retourne sous la forme d'une chaîne de caractères (format hexadécimal, en minuscule) le résultat de la fonction de hashage **MD5**. Elle peut également utiliser un autre algorithme de hashage (**SHA** par exemple). L'algorithme employé est spécifié dans le paramètre « algorithm ».

$KD(secret, data) = H(secret : data)$

La fonction KD appelle la fonction H avec comme argument la concaténation des deux paramètres secret et data séparés par le signe deux-points.

Le client envoie donc l'en-tête « Authorization » contenant le nom de la méthode « Digest » suivi des paramètres :

**username** Nom de l'utilisateur.

**realm** Même valeur que celle de la réponse du serveur.

**nonce** Même valeur que celle de la réponse du serveur.

**algorithm** Même valeur que celle de la réponse du serveur.

**opaque** Même valeur que celle de la réponse du serveur.

**uri** URI de la ressource protégée demandée (dupliquée ici car certains proxy peuvent modifier l'URI originale).

**response** Ce paramètre contenant 32 chiffres hexadécimaux représentant la valeur calculée par le client prouvant qu'il connaît le mot de passe.

**qop** (*optionnel*) (Quality Of Protection) Ce paramètre indique le niveau de protection appliqué. Il doit correspondre à l'une des valeurs retournées par le serveur.

**cnonce** (*si qop est présent*) Chaîne générée par le client.

**nc** (*si qop est présent*) (Nonce Count) 8 chiffres hexadécimaux représentant le nombre de fois que la valeur du paramètre « nonce » retournée par le serveur a été utilisée par le client. **nc=00000001** la première fois.

Le calcul de la valeur du paramètre response est effectué de la manière suivante :

Si qop est spécifié :

$response = KD( H(A1), nonce:nc:cnonce:qop:H(A2) )$

Sinon :

$response = KD( H(A1), nonce :H(A2) )$

Si algorithm vaut **MD5** ou n'est pas spécifié :

$A1 = username:realm:password$

si algorithm vaut **MD5-sess** :

$A1 = H(username:realm:password) :nonce:cnonce$

Si qop vaut **auth** ou n'est pas spécifié :

$A2 = http-method:uri$

si qop vaut **auth-int** :

$A2 = http-method:uri:H(entity)$

### 2.2.3 Réponse du serveur

Le serveur recalcule les mêmes valeurs que le client pour vérifier si l'identification est réussie.

Dans le cas où le serveur répond positivement (utilisateur et mot de passe corrects), il envoie, dans la réponse, l'en-tête HTTP « Authentication-Info » contenant des informations sur l'identification réussie et la prochaine identification.

Cet en-tête contient également une liste de paramètres séparés par une virgule :

**nextnonce** Valeur à utiliser pour les prochaines identifications dans ce domaine de protection.

**qop** (*Optionnel*) *quality of protection* appliquée à cette réponse. Ce paramètre doit avoir la même valeur que dans la requête du client.

**rspauth** (*Si qop spécifié*) Ce paramètre d'identification mutuel sert à prouver que le serveur connaît également l'utilisateur et son mot de passe. Il est calculé

de la même manière que le paramètre **response** excepté pour la valeur de A2 où http-method est une chaîne vide.

**cnonce** (*Si qop spécifié*) Même valeur que dans la requête du client.

**nc** (*Si qop spécifié*) Même valeur que dans la requête du client.

### 3 Identification sur serveur mandataire (*Proxy*)

L'identification décrite ci-dessus se déroule entre l'utilisateur et le serveur d'origine :

Il est également possible de s'identifier auprès des serveurs intermédiaires :

- Utilisateur à proxy
- Proxy à proxy
- Proxy à serveur d'origine.

Pour cela, les en-têtes **HTTP Proxy-Authenticate** et **Proxy-Authorization** sont utilisés à la place des en-têtes **WWW-Authenticate** et **Authorization**. Le code d'état **HTTP 407** est utilisé au lieu du code 401.

L'en-tête **Proxy-Authentication-Info** a le même rôle que l'en-tête **Authentication-Info**.

Un client peut devoir s'identifier à la fois à un proxy et au serveur d'origine, mais pas dans la même réponse.


L'identification ne peut être utilisée dans le cas d'un proxy transparent

## 4 Voir aussi

### 4.1 Articles connexes

- Hypertext Transfer Protocol
- Vulnérabilité des services d'authentification web

### 4.2 Liens externes

- (en) [RFC 2617] - *Authentication : Basic and Digest Access Authentication* - juin 1999
-  Portail de l'informatique

## 5 Sources, contributeurs et licences du texte et de l'image

### 5.1 Texte

- **Authentification HTTP** *Source* : [https://fr.wikipedia.org/wiki/Authentification\\_HTTP?oldid=112350121](https://fr.wikipedia.org/wiki/Authentification_HTTP?oldid=112350121) *Contributeurs* : Tieno, MedBot, Romanc19s, Sphinx, Shawn, AntonyB, Bouchecl, Le Pied-bot, DavidL, TXiKiBoT, Cimai, Chicobot, Cépey, MystBot, LordAnubisBOT, DumZiBoT, Lucas-bot, WolfyMoon, Nallimbot, Tanguy Ortol, MathsPoetry, Tango Panaché, KamikazeBot, EmausBot, Leodegar, Addbot, Margot Priem et Anonyme : 15

### 5.2 Images

- **Fichier:Crystal\_mycomputer.png** *Source* : [https://upload.wikimedia.org/wikipedia/commons/e/e3/Crystal\\_mycomputer.png](https://upload.wikimedia.org/wikipedia/commons/e/e3/Crystal_mycomputer.png) *Licence* : LGPL *Contributeurs* : All Crystal icons were posted by the author as LGPL on kde-look *Artiste d'origine* : Everaldo Coelho (YellowIcon);

### 5.3 Licence du contenu

- Creative Commons Attribution-Share Alike 3.0