Groupe de travail Réseau
Request for Comments : 2617
RFC rendue obsolète : 2069
Catégorie : Normes
juin 1999

Traduction Claude Brière de L'Isle, août 2007

J. Franks, Northwestern University
P. Hallam-Baker, Verisign, Inc.
J. Hostetler, AbiSource, Inc.
S. Lawrence, Agranat Systems, Inc.
P. Leach, Microsoft Corporation
A. Luotonen, Netscape Communications Corporation
L. Stewart, Open Market, Inc.

Authentification HTTP : Authentification d'accès Basic et Digest

Résumé
"HTTP/1.0", comporte la spécification d'un schéma d'authentification d'accès de base. Ce schéma n'est pas considéré comme une méthode sécurisée d'authentification d'utilisateur (sauf en conjonction avec un système externe sécurisé tel que SSL [RFC2046]) car le nom de l'utilisateur et le mot de passe sont passés en clair sur le réseau.

Le présent document fournit aussi la spécification du cadre d'authentification pour HTTP, le schéma original d'authentification de base et un schéma fondé sur des hachages cryptographiques, désignés sous le nom de "Digest Access Authentication" *(authentification d'accès par résumé)*. Il est donc aussi destiné à servir au remplacement de la [RFC2069]. Certains éléments facultatifs spécifiés dans la RFC 2069 ont été retirés de la présente spécification du fait de problèmes découverts depuis sa publication ; d'autres nouveaux éléments ont été ajoutés pour des questions de compatibilité, ces nouveaux éléments ont été rendus facultatifs, mais ils sont fortement recommandés.

Comme l'authentification de base, l'accès par résumé vérifie que les deux parties à une communication connaissent un secret partagé (un mot de passe) ; à la différence du Basic, cette vérification peut être effectuée sans envoyer le mot de passe en clair, ce qui est la plus grosse faiblesse du Basic. Comme avec la plupart des autres protocoles d'authentification, la plus grosse source de risques ne se trouve habituellement pas dans le protocole central lui-même mais dans les politiques et les procédures qui environnent son utilisation.

Table des matières

1 Authentification d'accès

1.1 Rôle de la spécification HTTP/1.1

La présente spécification est inséparable de la spécification de HTTP/1.1 [RFC2616]. Elle en utilise le paragraphe 2.1 sur le BNF augmenté, et s'appuie à la fois sur les non-terminaux définis dans ce document et sur d'autres aspects de la spécification HTTP/1.1.

1.2 Cadre de l'authentification d'accès

HTTP fournit un mécanisme d'authentification par simple interrogation réponse qui PEUT être utilisé par un serveur pour mettre en question une demande d'un client et par un client pour fournir des informations d'authentification. Il utilise un jeton *(token)* extensible, insensible à la casse pour identifier le schéma d'authentification, suivi par une liste de paires de valeurs d'attributs séparées par des virgules, qui porte les paramètres nécessaires pour réaliser l'authentification via ce schéma.

```
auth-scheme    = token
auth-param     = token "=" ( token | quoted-string )
```

Le message de réponse 401 (Non autorisé) est utilisé par un serveur d'origine pour mettre en cause l'autorisation d'un agent utilisateur. Cette réponse DOIT inclure un champ d'en-tête WWW-Authenticate contenant au moins une mise en cause applicable à la ressource demandée. Le message de réponse 407 (Authentification du mandataire exigée) est utilisé par un mandataire pour mettre en cause l'autorisation d'un client et DOIT inclure un champ d'en-tête Proxy-Authenticate contenant au moins une mise en cause applicable au mandataire pour la ressource demandée.

```
challenge      = auth-scheme 1*SP 1#auth-param
```

Note : Les agents utilisateurs devront faire particulièrement attention en analysant la valeur de champ d'en-tête de WWW-Authenticate ou de Proxy-Authenticate si il contient plus d'une mise en cause, ou si plus d'un champ d'en-tête WWW-Authenticate est fourni, car le contenu d'une mise en cause peut lui-même contenir une liste de paramètres d'authentification séparés par des virgules.

Le domaine des paramètres d'authentification est défini pour tous les schémas d'authentification :

```
realm     = "domaine" "=" valeur de domaine
realm-value    = chaîne en tre guillemets
```

La directive de domaine (insensible à la casse) est exigée pour tous les schémas d'authentification qui produisent une mise en cause. La valeur du domaine (insensible à la casse), combinée à l'URL racine canonique (l'URI absolu pour le serveur dont abs_path est vide ; voir le paragraphe 5.1.2 de [RFC2616]) du serveur auquel on accède, définit l'espace de protection. Ces domaines permettent de partager les ressources protégées d'un serveur en un ensemble d'espaces de

protection, ayant chacun son propre schéma d'authentification et/ou base de données d'autorisation. La valeur du domaine est une chaîne, généralement allouée par le serveur d'origine, qui peut avoir une sémantique supplémentaire, spécifique du schéma d'authentification. Noter qu'il peut y avoir plusieurs mises en cause avec le même schéma d'authentification mais des domaines différents.

Un agent utilisateur qui souhaite s'authentifier auprès d'un serveur d'origine (habituellement, mais pas nécessairement, après réception d'un message 401 (Non autorisé) PEUT le faire en incluant un champ d'en-tête Authorization dans la demande. Un client qui souhaite s'authentifier lui-même auprès d'un mandataire (habituellement, mais pas nécessairement, après avoir reçu un message 407 (Authentification de mandataire exigée) PEUT le faire en incluant un champ d'en-tête Proxy-Authorization dans la demande. La valeur du champ Authorization et la valeur du champ Proxy-Authorization consistent toutes deux en accréditifs contenant les informations d'authentification du client pour le domaine de la ressource demandée. L'agent utilisateur DOIT choisir d'utiliser une des mises en cause ayant le plus fort auth-scheme *(schéma d'authentification)* qu'il comprend et demander des accréditifs à l'utilisateur sur la base de cette mise en cause.

credentials = auth-scheme (token | qoted-string | #auth-param)

Noter que de nombreux navigateurs vont seulement reconnaître le Basic et vont demander que ce soit le premier auth-scheme présenté. Les serveurs ne devraient inclure le Basic que si il est au minimum acceptable.

L'espace de protection détermine le domaine sur lequel les accréditifs peuvent s'appliquer automatiquement. Si une demande antérieure avait été autorisée, les mêmes accréditifs PEUVENT être réutilisés pour toutes les autres demandes au sein de cet espace de protection pour une période déterminée par le schéma d'authentification, les paramètres, et/ou les préférences de l'utilisateur. Sauf définition contraire du schéma d'authentification, un seul espace de protection ne peut pas s'étendre au delà de la portée de son serveur.

Si le serveur d'origine ne souhaite pas accepter les accréditifs envoyés avec une demande, il DEVRAIT retourner une réponse 401 (Non autorisé). La réponse DOIT inclure un champ d'en-tête WWW-Authenticate contenant au moins une mise en cause (éventuellement nouvelle) applicable à la ressource demandée. Si un mandataire n'accepte pas les accréditifs envoyés avec une demande, il DEVRAIT retourner une réponse 407 (Authentification du mandataire exigée). La réponse DOIT inclure un champ d'en-tête Proxy-Authenticate contenant une mise en cause (éventuellement nouvelle) applicable au mandataire pour la ressource demandée.

Le protocole HTTP ne restreint pas les applications à ce simple mécanisme de mise en cause réponse pour l'authentification d'accès. Des mécanismes supplémentaires PEUVENT être utilisés, comme le chiffrement au niveau du transport ou via l'encapsulation du message, et avec des champs d'en-tête supplémentaires qui spécifient les informations d'authentification. Cependant, ces mécanismes supplémentaires ne sont pas définis par la présente spécification.

Les mandataires DOIVENT être complètement transparents en ce qui concerne l'authentification de l'agent utilisateur par les serveurs d'origine. C'est à dire qu'ils doivent transmettre les en-têtes WWW-Authenticate et Authorization inchangés, et suivre les règles qui figurent au paragraphe 14.8 de [RFC2616]. Les champs d'en-tête Proxy-Authenticate et Proxy-Authorization sont tous deux des en-têtes bond par bond (voir au paragraphe 13.5.1 de [RFC2616]).

2 Schéma d'authentification Basic

Le schéma d'authentification "basic" se fonde sur le modèle où le client doit s'authentifier avec un identifiant d'utilisateur *(user-ID)* et un mot de passe pour chaque domaine. La valeur du domaine devrait être considérée comme une chaîne opaque qui ne peut être comparée que pour égalité avec les autres domaines de ce serveur. Le serveur ne va servir la demande que si il peut valider le user-ID et le mot de passe pour l'espace de protection du Request-URI. Il n'y a pas de paramètre d'authentification facultatif.

Pour Basic, le cadre ci-dessus est utilisé comme suit :

challenge = "Basic" realm
credentials = "Basic" basic-credentials

À réception d'une demande non autorisée pour un URI au sein de l'espace de protection, le serveur d'origine PEUT répondre par une mise en question comme celle qui suit :

WWW-Authenticate: Basic realm="WallyWorld"

où "WallyWorld" est la chaîne allouée par le serveur pour identifier l'espace de protection de l'URI de demande. Un mandataire peut répondre avec la même mise en cause en utilisant le champ d'en-tête Proxy-Authenticate.

Pour recevoir l'autorisation, le client envoie le userid et le mot de passe, séparés par un seul caractère deux points (":"), au sein d'une chaîne codée en base64 [RFC2396] dans les accréditifs.

basic-credentials = base64-user-pass
base64-user-pass = <codage base64 [RFC2045] du mot de passe d'utilisateur, mais pas limité à 76 caractères/ligne>
user-pass = userid ":" mot de passe
userid = *<TEXT à l'exclusion de ":">
password = *TEXT

Le userid peut être insensible à la casse.

Si l'agent utilisateur souhaite envoyer le userid "Aladdin" et le mot de passe "open sesame", il devrait utiliser le champ d'en-tête suivant :

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

Un client DEVRAIT supposer que tous les chemins à la profondeur ou plus profonds que la profondeur du dernier élément symbolique dans le champ de chemin de l'URI de demande sont aussi au sein de l'espace de protection spécifié par la valeur de domaine Basic de la mise en cause en cours. Un client PEUT à titre conservatoire envoyer l'en-tête d'autorisation correspondant avec les demandes de ressources dans cet espace sans recevoir une autre mise en cause de la part du serveur. De même, lorsque un client envoie une demande à un mandataire, il peut réutiliser un userid et un mot de passe dans le champ d'en-tête Proxy-Authorization sans recevoir une autre mise en cause de la part du serveur mandataire. Voir à la section 4 les considérations sur la sécurité associées à l'authentification Basic.

3 Schéma d'authentification d'accès Digest

3.1 Introduction

3.1.1 Objet

Le protocole désigné comme "HTTP/1.0" inclut la spécification du schéma d'authentification d'accès Basic [RFC1945]. Ce schéma n'est pas considéré comme une méthode sûre pour l'authentification d'utilisateur, car le nom d'utilisateur et le mot de passe sont envoyés en forme non chiffrée sur le réseau. La présente section spécifie un schéma qui n'envoie pas le mot de passe en clair, qu'on appelle "Authentification d'accès Digest".

Le schéma d'authentification d'accès Digest n'est pas destiné à être une réponse complète au besoin de sécurité sur la Toile mondiale. Ce schéma ne fournit pas de chiffrement du contenu du message. L'intention est simplement de créer une méthode d'authentification d'accès qui évite les failles les plus sérieuses de l'authentification Basic.

3.1.2 Fonctionnement global

Comme dans l'authentification d'accès Basic, le schéma Digest se fonde sur un simple paradigme mise en cause/réponse. Le schéma Digest fait la mise en cause en utilisant une valeur de nom occasionnel *(nonce)*. Une réponse valide contient une somme de contrôle *(checksum)* (par défaut, la somme de contrôle MD5) du nom d'utilisateur, du mot de passe, de la valeur de nom occasionnel donné, de la méthode HTTP et de l'URI demandé. De cette façon, le mot de passe n'est jamais envoyé en clair. Tout comme avec le schéma Basic, le nom d'utilisateur et le mot de passe doivent être pré arrangés d'une certaine façon qui n'est pas traitée par ce document.

3.1.3 Représentation des valeurs résumées

Un en-tête facultatif permet au serveur de spécifier l'algorithme utilisé pour créer la somme de contrôle ou le résumé.

Par défaut, l'algorithme MD5 est utilisé et c'est le seul algorithme décrit dans le présent document.

Pour les besoins du présent document, un résumé MD5 de 128 bits est représenté par 32 caractères ASCII imprimables. Les bits du résumé de 128 bits sont convertis du bit de plus fort poids au bit de moindre poids, par groupes de quatre bits à la fois en leur représentation en ASCII comme suit. Chaque groupe de quatre bits est représenté par sa notation hexadécimale familière à partir des caractères 0123456789abcdef. C'est à dire que le binaire 0000 est représenté par le caractère '0', 0001, par '1', et ainsi de suite jusqu'à la représentation de 1111 par 'f'.

3.1.4 Limitations

Le schéma d'authentification Digest décrit dans le présent document souffre de nombreuses limitations connues. Il est destiné à remplacer l'authentification Basic et rien de plus. C'est un système fondé sur un mot de passe et (du côté du serveur) il souffre des mêmes problèmes que tout système de mot de passe. En particulier, aucune disposition n'est prévue dans ce protocole pour un arrangement sécurisé initial entre l'utilisateur et le serveur pour établir le mot de passe de l'utilisateur.

Les utilisateurs et les développeurs devraient être conscients que ce protocole n'est pas aussi sûr que Kerberos, et pas aussi sûr que n'importe quel schéma de clé privée du côté client. Néanmoins, c'est mieux que rien, et mieux que ce qui est actuellement utilisé avec telnet et ftp, et mieux que l'authentification Basic.

3.2 Spécification des en-têtes Digest

Le schéma d'authentification d'accès Digest est conceptuellement similaire au schéma Basic. Les formats de la ligne d'en-tête WWW-Authenticate modifiée et de la ligne d'en-tête Authorization sont spécifiés ci-dessous. De plus, un nouvel en-tête, Authentication-Info, est spécifié.

3.2.1 L'en-tête de réponse WWW-Authenticate

Si un serveur reçoit une demande d'objet protégé en accès, et si un en-tête Authorization acceptable n'est pas envoyé, le serveur répond par un code d'état "401 Non autorisé", et un en-tête WWW-Authenticate selon la trame définie ci-dessus, qui pour le schéma digest est utilisée comme suit :

```
challenge        =  "Digest" digest-challenge
digest-challenge = 1#( realm | [ domain ] | nonce | [ opaque ] |[ stale ] | [ algorithm ] |
                   [ qop-options ] | [auth-param] )
domain           = "domain" "=" <"> URI ( 1*SP URI ) <">
URI              = absoluteURI | abs_path
nonce            = "nonce" "=" nonce-value
nonce-value      = quoted-string
opaque           = "opaque" "=" quoted-string
stale            = "stale" "=" ( "true" | "false" )
algorithm        = "algorithm" "=" ( "MD5" | "MD5-sess" | token )
qop-options      = "qop" "=" <"> 1#qop-value <">
qop-value        = "auth" | "auth-int" | token
```

La signification des valeurs des directives utilisées ci-dessus est la suivante :

realm *(royaume)*
Une chaîne à afficher aux utilisateurs de sorte qu'ils sachent quel nom d'utilisateur *(username)* et mot de passe *(password)* utiliser. Cette chaîne devrait contenir au moins le nom de l'hôte qui effectue l'authentification et peut de plus indiquer la collection des utilisateurs qui peuvent avoir l'accès. Ce pourrait, par exemple être : "registered_users@gotham.news.com".

domain *(domaine)*
Liste d'URI entre guillemets, séparés par une espace, comme spécifié dans la RFC XURI [RFC2396], qui définit l'espace de protection. Si un URI est un abs_path, il se rapporte à l'URL racine canonique (voir au paragraphe 1.2 ci-dessus) du serveur d'accès. Un URI absolu dans cette liste peut se référer à un serveur différent de celui d'accès. Le client peut utiliser cette liste pour déterminer l'ensemble des URI pour lesquels les mêmes informations d'authentification peuvent être envoyées : tout URI qui a un URI comme préfixe dans cette liste (après que les deux ont

été rendus absolus) peut être supposé dans le même espace de protection. Si cette directive est omise ou si sa valeur est vide, le client devrait supposer que l'espace de protection consiste en tous les URI sur le serveur qui répond.

Cette directive n'est pas significative dans les en-têtes Proxy-Authenticate, pour lesquels l'espace de protection est toujours le mandataire entier ; si il est présent, il devrait être ignoré.

nonce *(nom occasionnel)*
Une chaîne de données spécifiée par le serveur devrait être générée de façon univoque chaque fois qu'est faite une réponse 401. Il est recommandé que cette chaîne soit en base64 ou en données hexadécimales. Précisément, comme la chaîne est passée dans les lignes d'en-tête comme une chaîne entre guillemets, le caractère doubles guillemets n'est pas admis.

Le contenu du nom occasionnel dépend de la mise en œuvre. La qualité de la mise en œuvre dépend d'un bon choix. Un nom occasionnel pourrait, par exemple, être construit comme le codage en base 64 de l'horodatage :

H(time-stamp ":" ETag ":" private-key)

où l'horodatage est l'heure générée par le serveur ou une autre valeur non répétitive, ETag est la valeur de l'en-tête ETag HTTP associé à l'entité demandée, et la clé privée sont des données qui ne sont connues que du serveur. Avec un nom occasionnel de cette forme, un serveur va recalculer la portion hachée après avoir reçu l'en-tête d'authentification du client et rejettera la demande si elle ne correspond pas au nom occasionnel de cet en-tête ou si la valeur de l'horodatage n'est pas assez récente. De cette façon, le serveur peut limiter la durée de validité du nom occasionnel. L'inclusion de l'ETag empêche une demande en répétition d'une version mise à jour de la ressource. (Note : inclure l'adresse IP du client dans le nom occasionnel apparaîtrait comme offrant au serveur la capacité à limiter la réutilisation du nom occasionnel au même client qui l'a eu à l'origine. Cependant, cela irait à l'encontre des locations de mandataires, pour lesquelles des demandes d'un seul utilisateur passent souvent par différents mandataires de la ferme. Et aussi, l'espionnage des adresses IP n'est pas si difficile.)

Une mise en œuvre peut choisir de ne pas accepter un nom occasionnel utilisé antérieurement, ou un résumé utilisé précédemment, afin de constituer une protection contre une attaque en répétition. Ou bien une mise en œuvre peut choisir d'utiliser des noms occasionnels ou résumés à utilisation unique pour des demandes POST ou PUT et un horodatage pour les demandes GET. Pour plus de précisions sur les problèmes que cela implique, voir la section 4 du présent document.

Le nom occasionnel ests opaque pour le client.

opaque
Une chaîne de données, spécifiée par le serveur, qui devrait être retournée inchangée par le client dans l'en-tête Authorization des demandes ultérieures avec des URI dans le même espace de protection. Il est recommandé que cette chaîne soit en base64 ou en données hexadécimales.

stale *(périmé)*
Fanion qui indique que la précédente demande du client a été rejetée parce que la valeur du nom occasionnel était périmée. Si stale est VRAI (insensible à la casse), le client peut souhaiter simplement réessayer la demande avec une nouvelle réponse chiffrée, sans réinitialiser l'utilisateur pour qu'il fournisse un nouveau nom d'utilisateur et mot de passe. Le serveur ne devrait mettre stale à VRAI que si il reçoit une demande pour laquelle le nom occasionnel est invalide mais avec un résumé valide pour ce nom occasionnel (ce qui indique que le client connaît le nom d'utilisateur/mot de passe correct). Si stale est FAUX, ou quelque chose d'autre que VRAI, ou si la directive stale n'est pas présente, le nom d'utilisateur et/ou le mot de passe sont invalides, et de nouvelles valeurs doivent être obtenues.

algorithm
Chaîne indiquant une paire d'algorithmes utilisés pour produire le résumé et une somme de contrôle. Si elle n'est pas présente, on suppose que c'est "MD5". Si l'algorithme n'est pas compris, la mise en cause doit être ignorée (et on doit en utiliser un autre, si il y en a plus d'un).

Dans le présent document, la chaîne obtenue en appliquant l'algorithme digest aux données "data" avec le secret "secret" sera notée KD(secret, data), et la chaîne obtenue en appliquant l'algorithme de somme de contrôle aux données data "data" sera notée H(data). La notation unq(X) signifie la valeur de la chaîne X entre guillemets sans les guillemets qui l'entourent.

Pour les algorithmes "MD5" et "MD5-sess"

H(data) = MD5(data)

et

KD(secret, data) = H(concat(secret, ":", data))

c'est-à-dire, le résumé est le MD5 du secret enchaîné avec deux points, enchaîné avec les données. L'algorithme "MD5-sess" est destiné à permettre des serveurs d'authentification tiers efficaces ; pour la différence d'utilisation, voir la description au paragraphe 3.2.2.2.

qop-options
Cette directive est facultative, mais ne l'est que pour la rétro compatibilité avec la [RFC2069] ; elle DEVRAIT être utilisée par toutes les mises en œuvre conformes à la présente version du schéma Digest. Si elle est présente, c'est une chaîne entre guillemets de un ou plusieurs jetons indiquant les valeurs de "qualité de protection" prises en charge par le serveur. La valeur "auth" indique l'authentification ; la valeur "auth-int" indique l'authentification avec protection d'intégrité ; voir les descriptions ci-dessous pour calculer la valeur de la directive de réponse pour l'application de ce choix. Les options non reconnues DOIVENT être ignorées.

auth-param
Cette directive permet des extensions futures. Toute directive non reconnue DOIT être ignorée.

3.2.2 L'en-tête de demande Authorization

Le client est supposé réessayer la demande, en passant une ligne d'en-tête Authorization, qui est définie conformément à la trame ci-dessus, utilisée comme suit :

```
credentials       = "Digest" digest-response
digest-response   = 1#( username | realm | nonce | digest-uri | response | [ algorithm ] | [cnonce] |
                    [opaque] | [message-qop] | [nonce-count] | [auth-param] )
username          = "username" "=" username-value
username-value    = quoted-string
digest-uri        = "uri" "=" <"> digest-uri-value <">
digest-uri-value  = request-uri   ; Comme spécifié par HTTP/1.1
message-qop       = "qop" "=" qop-value
cnonce            = "cnonce" "=" cnonce-value
cnonce-value      = nonce-value
nonce-count       = "nc" "=" nc-value
nc-value          = 8LHEX
response          = "response" "=" request-digest
request-digest    = <"> 32LHEX <">LHEX = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |"8" | "9" | "a" | "b" | "c"
                                        | "d" | "e" | "f"
```

Les valeurs des champs opaque et algorithm doivent être celles fournies dans l'en-tête de réponse WWW-Authenticate pour l'entité qui est demandée.

response
Chaîne de 32 chiffres hexadécimaux calculés comme défini ci-dessous, qui prouve que l'utilisateur connaît un mot de passe.

username
Le nom de l'utilisateur dans le domaine spécifié.

digest-uri
URI de l'URI de demande de la ligne de demande ; dupliquée ici parce que les mandataires sont autorisés à changer la ligne de demande dans le transit.

qop
Indique quelle "qualité de protection" le client a appliqué au message. Si elle est présente, sa valeur DOIT être une des solutions de remplacement que le serveur a indiqué accepter dans l'en-tête WWW-Authenticate. Ces valeurs affectent le calcul du résumé de la demande. Noter qu'il s'agit d'un seul jeton, et non d'une liste entre guillemets des solutions

de remplacement comme dans WWW-Authenticate. Cette directive est facultative afin de préserver la rétro compatibilité avec une mise en œuvre minimale de la [RFC2069], mais DEVRAIT être utilisée si le serveur a indiqué que qop est pris en charge en fournissant une directive qop dans le champ d'en-tête WWW-Authenticate.

cnonce
Il DOIT être spécifié si une directive qop est envoyée (voir ci-dessus), et NE DOIT PAS être spécifié si le server n'a pas envoyé de directive qop dans le champ d'en-tête WWW-Authenticate. La valeur du cnonce est une valeur de chaîne opaque entre guillemets fournie par le client et utilisée à la fois par le client et le serveur pour éviter les attaques contre le texte en clair, pour fournir l'authentification mutuelle et avoir une certaine protection de l'intégrité du message. Voir ci-dessous les descriptions du calcul des valeurs de response-digest et request-digest.

nonce-count
Il DOIT être spécifie si une directive qop est envoyée (voir ci-dessus), et NE DOIT PAS être spécifié si le serveur n'a pas envoyé de directive qop dans le champ d'en-tête WWW-Authenticate. La valeur de nc est le compte hexadécimal du nombre de demandes (y compris la demande en cours) que le client a envoyé avec la valeur du nom occasionnel dans cette demande. Par exemple, dans la première demande envoyée en réponse à une valeur de nom occasionnel donné, le client envoie "nc=00000001". L'objet de cette directive est de permettre au serveur de détecter les répétitions de demande en entretenant sa propre copie du compte – si la même valeur nc est envoyée deux fois, la demande est alors une répétition. Voir ci-dessous la description de la construction de la valeur du résumé de la demande *(request-digest)*.

auth-param
Cette directive permettra des extensions à l'avenir. Toute directive non reconnue DOIT être ignorée.

Si une directive ou sa valeur est impropre, ou si les directives nécessaires manquent, la réponse appropriée est 400 Mauvaise demande. Si le résumé de demande est invalide, on devrait alors enregistrer une défaillance de connexion, car des défaillances de connexion de la part d'un même client peuvent indiquer qu'un attaquant essaye de deviner des mots de passe.

La définition du résumé de demande ci-dessus indique le codage de sa valeur. Les définitions qui suivent montent la façon dont cette valeur est calculée.

3.2.2.1 Request-Digest *(résumé de demande)*

Si la valeur du "qop" est "auth" ou "auth-int" :

request-digest = <"> < KD (H(A1), unq(valeur du nom occasionnel) ":" nc-value ":" unq(valur du cnonce) ":" unq(valeur de qop) ":" H(A2)) > <">

Si la directive "qop" n'est pas présente (cette construction est donnée pour la compatibilité avec la RFC 2069) :

request-digest = <"> < KD (H(A1), unq(valeur du nom occasionnel) ":" H(A2)) > <">

Voir ci-dessous les définitions de A1 et A2.

3.2.2.2 A1

Si la valeur de la directive "algorithm" est "MD5" ou n'est pas spécifiée, alors A1 est :

A1 = unq(valeur du nom d'utilisateur) ":" unq(valeur du domaine) ":" passwd

où

passwd = < mot de passe d'utilisateur >

Si la valeur de la directive "algorithm" est "MD5-sess", alors A1 n'est calculé qu'une seule fois – sur la première demande du client qui suit la réception d'une mise en cause WWW-Authenticate de la part du serveur. Il utilise le nom occasionnel du serveur à partir de cette mise en cause, et la première valeur de nom occasionnel de client pour construire A1 comme suit :

A1 = H(unq(valeur de nom d'utilisateur) ":" unq(valeur de domaine) ":" passwd)

":" unq(valeur de nom occasionnel) ":" unq(valeur de cnonce)

Ceci crée une 'clé de session' pour l'authentification des demandes et réponses ultérieures qui sont différentes pour chaque "session d'authentification", limitant donc la quantité de matériel haché avec n'importe laquelle des clés.
(Note : voir plus loin au paragraphe 3.3 l'exposé sur la session d'authentification.) Comme le serveur a seulement besoin d'utiliser le hachage des accréditifs d'utilisateur pour créer la valeur de A1, cette construction pourrait être utilisée en conjonction avec un service d'authentification de tiers de sorte que le serveur de la toile n'ait pas besoin de la valeur réelle du mot de passe. La spécification d'un tel protocole sort du domaine d'application de la présente spécification.

3.2.2.3 A2

Si la valeur de la directive "qop" est "auth" ou n'est pas spécifiée, alors A2 est :
A2 = Method ":" digest-uri-value

Si la valeur de "qop" est "auth-int", alors A2 est :
A2 = Method ":" digest-uri-value ":" H(corps-d'entiy)

3.2.2.4 Valeurs et chaînes entre guillemets de directive

Noter que la valeur de beaucoup de directives, telles que "username-value", est définie comme une "quoted-string" *(chaîne entre guillemets)*. Cependant, la notation "unq" indique que les guillemets qui l'entourent sont retirés lors de la formation de la chaîne A1. Et donc, si l'en-tête Authorization inclut les champs username="Mufasa", realm=myhost@testrealm.com et si l'utilisateur Mufasa a un mot de passe de "Circle Of Life" H(A1) devrait alors être H(Mufasa:myhost@testrealm.com:Circle Of Life) sans guillemets dans la chaîne résumée.

Aucune espace blanche n'est permise dans les chaînes auxquelles est apliquée la fonction H() de résumé sauf si cette espace blanche existe dans les chaînes entre guillemets ou dans le corps d'entité dont le contenu constitue la chaîne à résumer. Par exemple, la chaîne A1 illustrée ci-dessus doit être :

Mufasa:myhost@testrealm.com:Circle Of Life

sans espace ni d'un côte ni de l'autre des deux points, mais avec une espace entre les mots utilisés dans la valeur du mot de passe. De même, les autres chaînes résumées par H() ne doivent pas avoir d'espace d'un côté ni de l'autre des deux points qui délimitent leurs champs sauf si cette espace blanche figurait dans les chaînes entre guillemets ou corps d'entité à résumer.

Noter aussi que si on applique la protection d'intégrité (qop=auth-int), le H(corps-d'entité) est le hachage du corps d'entité, et non le corps du message – il est calculé avant l'application de tout codage de transfert par l'envoyeur et après qu'il a été retiré par le receveur. Noter que cela inclut des frontières multi-parties et des en-têtes incorporés dans chaque partie de tout type de contenu multi-parties.

3.2.2.5 Considérations diverses

La valeur "Method" est la méthode de demande HTTP comme spécifié au paragraphe 5.1.1 de la [RFC2616]. La valeur de "request-uri" est l'URI de demande de la ligne de demande, comme spécifié au paragraphe 5.1.2 de la [RFC2616]. Elle peut être "*", un "URL absolu" ou un "abs_path" comme spécifié au paragraphe 5.1.2 de la [RFC2616], mais elle DOIT être en accord avec l'URI de demande. En particulier, elle DOIT être un "absoluteURL" si l'URI de demande est un "absoluteURL". La "valeur de cnonce" est une valeur facultative choisie par le client, dont l'objet est de déjouer les attaques par texte en clair choisi.

Le serveur d'authentification doit s'assurer que la ressource désignée par la directive "uri" est la même que celle spécifiée dans la ligne de demande ; si elle ne l'est pas, le serveur DEVRAIT retourner une erreur 400 Mauvaise demande. (Comme ce peut être le symptôme d'une attaque, les développeurs de serveurs peuvent vouloir envisager d'enregistrer de telles erreurs.) L'idée derrière la duplication des informations de l'URL de demande dans ce champ est de faire face à la possibilité qu'un mandataire intermédiaire altère la ligne de demande du client. Cette demande altérée (mais qu'on peut supposer sémantiquement équivalente) ne résulterait pas en un résumé identique à celui calculé par le client.

Le développeurs devraient être conscient de l'interaction des transactions authentifiées avec les antémémoires partagées. Le protocole HTTP/1.1 spécifie que lorsqu'une antémémoire partagée (voir au paragraphe 13.7 de la

[RFC2616]) a reçu une demande contenant un en-tête Authorization et une réponse relayant cette demande, il NE DOIT PAS retourner cette réponse comme réplique à aucune autre demande, sauf si une des deux directives Cache-Control (voir au paragraphe 14.9 de la [RFC2616]) est présente dans la réponse. Si la réponse d'origine incluait la directive de commande d'antémémoire "must-revalidate", l'antémémoire PEUT utiliser l'entité de cette réponse en répondant à une demande ultérieure, mais DOIT d'abord la revalider auprès du serveur d'origine, en utilisant les en-têtes de la demande tirés de la nouvelle demande pour permettre au serveur d'origine d'authentifier la nouvelle demande. Autrement, si la réponse d'origine incluait la directive de commande d'antémémoire "public", l'entité de réponse PEUT être retournée en réplique à toute demande ultérieure.

3.2.3 L'en-tête Authentication-Info

L'en-tête Authentication-Info est utilisé par le serveur pour communiquer des informations concernant le succès de l'authentification dans la réponse.

```
AuthenticationInfo  = "Authentication-Info" ":" auth-info
auth-info           = 1#(nextnonce | [ message-qop ] | [ response-auth ] | [ cnonce ] | [nonce-count] )
nextnonce           = "nextnonce" "=" valeur de nom occasionnel
response-auth       = "rspauth" "=" résumé de réponse
response-digest     = <"> *LHEX <">
```

La valeur de la directive nextnonce est le nom occasionnel que le serveur souhaite que le client utilise pour une future réponse d'authentification. Le serveur peut envoyer l'en-tête Authentication-Info avec un champ nextnonce *(prochain nom occasionnel*) comme moyen de mettre en œuvre des noms occasionnels à utilisation unique ou changeants d'une autre façon. Si le champ nextnonce est présent, le client DEVRAIT l'utiliser lorsqu'il construit l'en-tête Authorization pour sa prochaine demande. L'échec du client à le faire peut avoir pour résultat qu'une demande de ré-authentification de la part du serveur aura "stale=VRAI".

Les mises en œuvre de serveur devraient considérer avec attention les implications sur les performances de l'utilisation de ce mécanisme ; les demandes tunnelées ne seront plus possibles si chaque réponse comporte une directive nextnonce qui doit être utilisée sur la prochaine demande reçue par le serveur. Il faudra considérer le compromis entre performance et sécurité pour permettre l'utilisation d'une vieille valeur de nom occasionnel pendant une période limitée afin d'autoriser le tunnelage de demandes. L'utilisation d'un compte de nom occasionnel peut préserver la plus grande partie des avantages de la sécurité d'un nouveau nom occasionnel de serveur sans effets délétères sur le tunnelage.

message-qop
Indique les options de "qualité de protection" appliquées à la réponse par le serveur. La valeur "auth" indique l'authentification ; la valeur "auth-int" indique l'authentification avec protection de l'intégrité. Le serveur DEVRAIT utiliser la même valeur pour la directive message-qop dans la réponse que celle envoyée par le client dans la demande correspondante.

Le résumé de réponse facultatif dans la directive "response-auth" prend en charge l'authentification mutuelle -- le serveur prouve qu'il connaît le secret de l'utilisateur, et avec qop=auth-int fournit aussi une protection d'intégrité limitée de la réponse. La valeur "résumé de réponse" est calculée comme pour le "résumé de demande" dans l'en-tête Authorization, sauf que si "qop=auth" ou n'est pas spécifié dans l'en-tête Authorization pour la demande, A2 est

```
A2      = ":" digest-uri-value
```

et si "qop=auth-int", alors A2 est

```
A2      = ":" digest-uri-value ":" H(corps d'entité)
```

où "digest-uri-value" est la valeur de la directive "uri" sur l'en-tête Authorization de la demande. La "cnonce-value" et la "nc-value" DOIVENT être celles de la demande du client à laquelle ce message est la réponse. Les directives "response-auth", "cnonce", et "nonce-count" DOIVENT être présentes si "qop=auth" ou "qop=auth-int" est spécifié.

L'en-tête Authentication-Info est permis dans l'en-queue d'un message HTTP transféré via un codage de transfert tronqué.

3.3 Fonctionnement de Digest

À réception de l'en-tête Authorization, le serveur peut vérifier sa validité en cherchant le mot de passe qui correspond au nom d'utilisateur soumis. Puis, le serveur doit effectuer la même opération de résumé (par exemple, MD5) effectuée par le client, et comparer le résultat à la valeur du résumé-de-demande donné.

Noter que le serveur HTTP n'a pas réellement besoin de connaître le mot de passe en clair de l'utilisateur. Tant que H(A1) est disponible pour le serveur, la validité d'un en-tête Authorization peut être vérifiée.

La réponse du client à une mise en cause WWW-Authenticate pour un espace de protection commence une session d'authentification avec cet espace de protection. La session d'authentification dure jusqu'à ce que le client reçoive une autre mise en cause WWW-Authenticate de tout serveur dans l'espace de protection. Un client devrait se souvenir du nom d'utilisateur, du mot de passe, du nom occasionnel, du compte de nom occasionnel et des valeurs opaques associées à une session d'authentification à utiliser pour construire l'en-tête Authorization dans les demandes futures au sein de cet espace de protection. L'en-tête Authorization peut être inclus de façon préalable ; le faire améliore l'efficacité du serveur et évite des allers-retours supplémentaires pour les mises en causes d'authentification. Le serveur peut choisir d'accepter les anciennes informations d'en-tête Authorization, quand bien même la valeur du nom occasionnel inclus ne serait plus fraîche. Autrement, le serveur peut retourner une réponse 401 avec une nouvelle valeur de nom occasionnel, amenant le client à réessayer la demande, en spécifiant stale=VRAI avec cette réponse, le serveur dit au client de réessayer avec le nouveau nom occasionnel, mais sans exiger un nouveau nom d'utilisateur et un nouveau mot de passe.

Comme le client est obligé de retourner la valeur de la directive opaque qui lui a été donnée par le serveur pour la durée d'une session, les données opaques peuvent être utilisées pour transporter les informations d'état de la session d'authentification. (Noter qu'un tel résultat peut être atteint plus facilement et plus sûrement en incluant l'état dans le nom occasionnel.) Par exemple, un serveur pourrait être chargé d'authentifier un contenu qui réside en fait sur un autre serveur. Il le ferait en incluant dans la première réponse 401 une directive de domaine dont la valeur inclurait un URI sur le second serveur, et une directive opaque dont la valeur contiendrait les informations d'état. Le client réessayera la demande, et à ce moment, le serveur pourra répondre par une redirection 301/302, pointant sur l'URI du second serveur. Le client suivra la redirection, et passera un en-tête Authorization incluant les données <opaque>.

Comme avec le schéma de base, les mandataires doivent être complètement transparents dans le schéma d'authentification d'accès Digest. C'est-à-dire qu'ils doivent transmettre les en-têtes WWW-Authenticate, Authentication-Info et Authorization inchangés. Si un mandataire veut authentifier un client avant la transmission d'une demande au serveur, il peut le faire en utilisant les en-têtes Proxy-Authenticate et Proxy-Authorization décrits au paragraphe 3.6 ci-après.

3.4 Négociation du protocole de sécurité

Il est utile qu'un serveur soit capable de savoir quels schémas de sécurité un client a la possibilité de traiter.

Il est possible qu'un serveur veuille exiger Digest comme sa méthode d'authentification, même si le serveur ne sait pas si le client l'accepte. Un client est invité à renoncer de bonne grâce si le serveur ne spécifie que des schémas d'authentification qu'il ne peut pas traiter.

3.5 Exemple

L'exemple suivant suppose qu'un document en accès protégé est demandé à partir d'un serveur via une demande GET. L'URI du document est "http://www.nowhere.org/dir/index.html". Le client et le serveur savent tous deux que le nom d'utilisateur pour ce document est "Mufasa", et que le mot de passe est "Circle Of Life" (avec une espace entre chacun des trois mots).

La première fois que le client demande le document, aucun en-tête Authorization n'est envoyé, aussi le serveur répond par :

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
        realm="testrealm@host.com",
        qop="auth,auth-int",
```

```
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Le client peut inviter l'utilisateur à fournir le nome d'utilisateur et le mot de passe, après quoi il va répondre par une nouvelle demande, incluant l'en-tête Authorization suivant :

```
Authorization: Digest username="Mufasa",
        realm="testrealm@host.com",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        uri="/dir/index.html",
        qop=auth,
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

3.6 Proxy-Authentication et Proxy-Authorization

Le schéma d'authentification par résumé peut aussi être utilisé pour authentifier les utilisateurs auprès des mandataires, les mandataires auprès des mandataires, ou les mandataires auprès des serveurs d'origine, en utilisant les en-têtes Proxy-Authenticate et Proxy-Authorization. Ces en-têtes sont des instances des en-têtes Proxy-Authenticate et Proxy-Authorization spécifiés aux paragraphes 14.33 et 14.34 de la spécification HTTP/1.1 [RFC2616] et leur comportement est soumis aux restrictions qui y sont décrites. Les transactions pour l'authentification de mandataires sont très similaires à celles déjà décrites. À réception d'une demande qui requiert l'authentification, le mandataire/serveur doit produire la réponse "407 Authentification du mandataire exigée" avec un en-tête "Proxy-Authenticate". La mise en cause de résumé utilisée dans l'en-tête Proxy-Authenticate est la même que pour l'en-tête WWW-Authenticate comme défini ci-dessus au paragraphe 3.2.1.

Le client/mandataire doit alors refaire la demande avec un en-tête Proxy-Authorization, avec les directives spécifiées pour l'en-tête Authorization au paragraphe 3.2.2 ci-dessus.

Dans les réponses ultérieures, le serveur envoie les Proxy-Authentication-Info avec les mêmes directives que pour le champ d'en-tête Authentication-Info.

Noter qu'en principe, il pourrait être demandé à un client de s'authentifier à la fois auprès d'un mandataire et d'un serveur terminal, mais jamais dans la même réponse.

4 Considérations sur la sécurité

4.1 Authentification des clients à l'aide de l'authentification Basic

Le schéma d'authentification Basic n'est pas une méthode sûre d'authentification d'utilisateur, et ne protège en aucune façon l'entité qui est transmise en clair à travers le réseau physique utilisé comme transporteur. HTTP n'empêche pas d'utiliser des schémas d'authentification et des mécanismes de chiffrement supplémentaires pour accroître la sécurité ou ajouter des améliorations (comme des schémas utilisant des mots de passe à usage unique) à l'authentification de base.

La faiblesse la plus sérieuse de l'authentification de base est qu'elle résulte en la transmission essentiellement en clair du mot de passe de l'utilisateur sur le réseau physique. C'est ce problème que l'authentification par résumé essaye de régler.

Comme l'authentification de base implique la transmission en clair des mots de passe, elle NE DEVRAIT PAS être utilisée (sans améliorations) pour protéger des informations sensibles ou précieuses.

Une utilisation courante de l'authentification de base est faite pour des besoins d'identification – exigeant que l'utilisateur fournisse un nom d'utilisateur et un mot de passe comme moyens d'identification, par exemple, pour rassembler des statistiques d'utilisation précises sur un serveur. Utilisée de cette façon, il est tentant de penser qu'il n'y a aucun danger si l'accès illicite aux documents protégés n'est pas un sujet de préoccupation majeur. Ceci n'est correct

que si le serveur produit à la fois le nom d'utilisateur et le mot de passe aux utilisateurs et en particulier, ne permet pas à l'utilisateur le choix de son propre mot de passe. Le danger vient de ce que l'utilisateur naïf réutilise fréquemment un seul et même mot de passe pour éviter d'avoir à se souvenir de plusieurs mots de passe.

Si un serveur permet aux utilisateurs de choisir leurs propres mots de passe, la menace n'est plus seulement l'accès non autorisé aux documents du serveur mais aussi l'accès non autorisé à toutes les autres ressources sur les autres systèmes que l'utilisateur protège avec le même mot de passe. De plus, dans la base de donnée de mots de passe du serveur, de nombreux mots de passe peuvent aussi être des mots de passe d'utilisateurs pour d'autres sites. Le propriétaire ou administrateur d'un tel système peut donc exposer tous les utilisateurs du système au risque d'accès non autorisé à tous les sites si ces informations ne sont pas conservées de façon sécurisée.

L'authentification de base est aussi vulnérable à l'espionnage par de faux serveurs. Si un utilisateur peut être conduit à croire qu'il se connecte à un hôte contenant des informations protégées par l'authentification de base, alors qu'en fait, il se connecte à un serveur ou une passerelle hostile, l'attaquant peut alors demander un mot de passe, le mémoriser pour une occasion ultérieure, et feindre une erreur. Ce type d'attaque n'est pas possible avec l'authentification par résumé *(Digest)*. Les développeurs de serveurs DEVRAIENT être en garde contre la possibilité de cette sorte de contrefaçon par des passerelles ou scripts CGI. En particulier, il est très dangereux pour un serveur de simplement fermer une connexion avec une passerelle. Cette passerelle peut alors utiliser le mécanisme persistant de connexion pour s'engager dans plusieurs transactions avec le client tout en se faisant passer pour le serveur d'origine d'une façon qui n'est pas détectable par le client.

4.2 Authentification des clients à l'aide de l'authentification Digest

L'authentification Digest ne donne pas un mécanisme d'authentification fort, comparé, par exemple, aux mécanismes à base de clés publiques. Cependant, il est significativement plus fort que (par exemple) CRAM-MD5, qu'il a été proposé d'utiliser avec LDAP [RFC2829], POP et IMAP (voir la [RFC2195]). Il est destiné à remplacer le mécanisme Basic, bien plus faible et même dangereux.

L'authentification par résumé *(Digest)* n'offre aucune protection de la confidentialité en dehors de la protection du mot de passe réel. Tout le reste de la demande et de la réponse est disponible pour les oreilles indiscrètes.

L'authentification par résumé n'offre qu'une protection limitée de l'intégrité pour les messages dans l'une ou l'autre direction. Si le mécanisme qop=auth-int est utilisé, les parties du message utilisées dans le calcul des valeurs de directive de champ d'en-tête WWW-Authenticate et Authorization (voir au paragraphe 3.2 ci-dessus) sont protégées. La plupart des champs d'en-tête et leurs valeurs pourraient être modifiées au titre d'une attaque par interposition.

De nombreux besoins de sécurisation des transactions HTTP ne peuvent pas être satisfaits par l'authentification Digest. Pour ces besoins, TLS ou SHTTP sont des protocoles plus appropriés. En particulier l'authentification par résumé ne peut pas être utilisée pour des transactions qui exigent une protection de la confidentialité. Néanmoins, de nombreuses fonctions restent, pour lesquelles l'authentification par résumé est à la fois utile et appropriée. Tout service utilisé à présent qui se sert de Basic devrait être basculé sur Digest aussi vite que possible.

4.3 Valeurs de nom occasionnel à utilisation limitée

Le schéma Digest utilise un nom occasionnel spécifié par le serveur pour servir de germe à la génération de la valeur de résumé de la demande (comme spécifié au paragraphe 3.2.2.1 ci-dessus). Comme indiqué dans l'exemple de nom occasionnel du paragraphe 3.2.1, le serveur est libre de construire le nom occasionnel de telle sorte qu'il ne puisse être utilisé que par un client particulier, pour une ressource particulière, pour une durée limitée ou un nombre d'utilisations limité, ou toutes autres restrictions. Faire ainsi renforce la protection accordée contre, par exemple, les attaques en répétition (voir au paragraphe 4.5). Cependant, il vaut de noter que la méthode choisie pour générer et vérifier le nom occasionnel a aussi des implications sur les performances et les ressources. Par exemple, un serveur peut choisir de permettre à chaque nom occasionnel de n'être utilisé qu'une seule fois en mémorisant si chaque nom occasionnel récemment produit a été ou non retourné et en envoyant une directive next-nonce *(prochain nom occasionnel)* dans le champ d'en-tête Authentication-Info de chaque réponse. Ceci protège même contre une attaque en répétition immédiate, mais a un coût élevé en vérification des valeurs de nom occasionnel, et, peut-être plus important, va causer des échecs d'authentification pour toute demande tunnelée (présumée retourner une indication périmée de nom occasionnel). De même, incorporer un élément spécifique d'une demande tel qu'une valeur ETag pour une ressource limite l'utilisation du nom occasionnel à cette version de la ressource et met aussi à mal le tunnelage. Et donc il peut être utile de faire ainsi pour les méthodes qui ont des effets latéraux mais ont des performances inacceptables pour celles qui n'en ont pas.

4.4 Comparaison de l'authentification Digest et Basic

Les authentifications Digest et Basic sont toutes deux sur le côté faible du spectre de force de la sécurité. Mais une comparaison entre les deux montre l'utilité, et même la nécessité, de remplacer Basic par Digest.

La plus grande menace sur le type de transactions pour lesquelles ces protocoles sont utilisés est l'espionnage du réseau. Cette sorte de transaction peut impliquer, par exemple, l'accès en ligne à une base de données dont l'utilisation est réservée à des abonnés payants. Avec l'authentification de base, un espion peut obtenir le mot de passe de l'usager. Cela ne lui permet pas seulement l'accès à tout de qui est dans la base de données, mais, souvent pire, lui permet l'accès à toutes les autres choses que l'usager protège avec le même mot de passe.

À l'inverse, avec l'authentification par résumé, l'espion n'obtiendra l'accès qu'à la transaction en question et pas au mot de passe de l'usager. Les informations obtenues par l'espion permettraient une attaque en répétition, mais seulement avec une demande pour le même document, et même cela peut être limité par le choix du nom occasionnel du serveur.

4.5 Attaques en répétition

Une attaque en répétition contre l'authentification par résumé va normalement être sans objet pour une simple demande GET car un espion aura déjà vu le seul document qu'il pourrait obtenir par une répétition. Ceci parce que l'URI du document demandé est résumé dans la demande du client et que le serveur ne livrera que ce document. Au contraire, dans l'authentification de base, un fois que l'espion a le mot de passe de l'usager, tout document protégé par ce mot de passe lui est ouvert.

Et donc, pour certains objets, il est nécessaire de protéger contre les attaques en répétition. Une bonne mise en œuvre de Digest peut le faire de diverses façons. La valeur de "nom occasionnel" créée par le serveur dépend de la mise en œuvre, mais si elle contient un résumé de l'adresse IP du client, un horodatage, l'ETag de la ressource, et une clé privée de serveur (comme recommandé ci-dessus) une attaque en répétition n'est alors pas simple. Un attaquant doit convaincre le serveur que la demande vient d'une fausse adresse IP et doit amener le serveur à livrer le document à une adresse IP différente de l'adresse à laquelle il croit qu'il envoie le document. Une attaque ne peut réussir que dans la période précédant l'expiration de l'horodatage. Résumer l'adresse IP du client et l'horodatage dans le nom occasionnel permet une mise en œuvre qui ne maintient pas l'état entre les transactions.

Pour les applications où aucune possibilité d'attaque en répétition ne peut être tolérée, le serveur peut utiliser des valeurs de nom occasionnel à utilisation unique, qui ne seront pas honorées pour un second usage. Cela exige que l'enveloppe du serveur se souvienne des valeurs de nom occasionnel qui ont été utilisées jusqu'à ce que l'horodatage du nom occasionnel (et donc le résumé construit avec lui) soir arrivé à expiration, mais il protège effectivement contre les attaques en répétition.

Une mise en œuvre doit tout particulièrement veiller aux possibilités d'attaques en répétition avec les demandes POST et PUT. Sauf si le serveur emploie des noms occasionnels à utilisation unique ou limitées par d'autres moyens et/ou insiste pour que soit utilisée la protection d'intégrité de qop=auth-int, un attaquant peut répéter des accréditifs valides à partir d'une demande réussie avec des données contrefaites ou un autre corps de message. Même avec l'utilisation de la protection d'intégrité, la plupart des métadonnées dans les champs d'en-tête ne sont pas protégées. Une génération et une vérification appropriée du nom occasionnel donne une certaine protection contre les attaques en répétition d'accréditifs valides précédemment utilisés, mais voir au paragraphe 4.8.

4.6 Faiblesses créées par plusieurs schémas d'authentification

Un serveur HTTP/1.1 peut retourner plusieurs mises en cause avec une réponse 401 (Authentifier) et chaque mise en cause peut utiliser un schéma d'authentification différent. Un agent utilisateur DOIT choisir d'utiliser le plus fort schéma d'authentification qu'il comprend et demander les accréditifs à l'usager sur la base de cette mise en cause.

Noter que de nombreux navigateurs vont ne reconnaître que l'authentification Basic et vont demander que ce soit le premier schéma d'authentification présenté. Les serveurs devraient n'inclure le Basic que si il est le minimum acceptable.

Lorsque le serveur offre le choix des schémas d'authentification en utilisant l'en-tête WWW-Authenticate, la force de l'authentification résultante n'est qu'aussi bonne que le plus faible des schémas d'authentification. Voir au paragraphe 4.8 ci-dessous l'exposé sur des scénarios particuliers d'attaque qui exploitent plusieurs schémas d'authentification.

4.7 Attaques de dictionnaire en ligne

Si l'attaquant peut espionner, il peut alors essayer toutes les paires de nom occasionnel/réponse entendues contre une liste de mots courants. Une telle liste est habituellement bien plus petite que le nombre total de mots de passe possibles. Le coût de calcul de la réponse pour chaque mot de passe sur la liste est payé une fois pour chaque mise en cause.

Le serveur peut atténuer cette attaque en ne permettant pas aux usagers de choisir des mots de passe qui soient dans un dictionnaire.

4.8 Attaque par interposition

Les authentifications Basic et Digest sont toutes deux vulnérables aux attaques par un intrus interposé (MITM, *man in the middle*) par exemple, de la part d'un mandataire hostile ou compromis. Cela représente très clairement tous les problèmes de l'espionnage. Mais cela offre aussi quelques opportunités supplémentaires à l'attaquant.

Une attaque par interposition possible serait d'ajouter un schéma d'authentification faible à l'ensemble des choix, en souhaitant que le client en utilise un qui expose les accréditifs de l'usager (par exemple, le mot de passe). Pour cette raison, le client devrait toujours utiliser le schéma le plus fort qu'il comprend dans les choix offerts.

Une bien meilleure attaque MITM serait de retirer tous les choix offerts, et de les remplacer par une mise en cause qui ne demande que l'authentification Basic, puis utilise les accréditifs en clair tirés de l'authentification Basic pour s'authentifier auprès du serveur d'origine en utilisant le plus fort schéma qu'il demande. Une façon particulièrement insidieuse de monter une telle attaque MITM serait d'offrir un service de mandataire d'antémémoire "gratuit" à des utilisateurs crédules.

Les agents d'utilisateur devraient envisager des mesures telles que la présentation d'une indication visuelle au moment de la demande d'accréditifs sur le schéma d'authentification qui est à utiliser, ou rappelant le plus fort schéma d'authentification demandé par le serveur et produisant un message d'avertissement avant d'en utiliser un plus faible. Ce pourrait également être une bonne idée pour l'agent d'utilisateur d'être configuré pour demander en général l'authentification Digest, ou à partir de sites spécifiques. Ou bien, un mandataire hostile peut mystifier le client en faisant une demande voulue par l'attaquant plutôt que celle voulue par le client. Bien sûr, ceci est quand même plus dur qu'une attaque comparable contre une authentification de base.

4.9 Attaques par texte en clair choisi

Avec l'authentification par résumé, un MITM ou un serveur malveillant peut arbitrairement choisir le nom occasionnel que le client va utiliser pour calculer la réponse. C'est ce qu'on appelle une attaque par "texte en clair choisi". La capacité à choisir le nom occasionnel est connue pour rendre l'analyse cryptographique plus facile [MD5].

Cependant, on ne connaît actuellement aucun moyen d'analyser la fonction unidirectionnelle MD5 utilisée par Digest en utilisant le texte en clair choisi.

La contre mesure des clients contre cette attaque est d'être configuré pour demander l'utilisation de la directive "cnonce" facultative ; cela permet au client de varier l'entrée du hachage d'une façon qui ne soit pas choisie par l'attaquant.

4.10 Attaques de dictionnaire pré calculé

Avec l'authentification Digest, si l'attaquant peut exécuter une attaque de texte en clair choisi, il peut pré calculer la réponse pour de nombreux mots courants à un nom occasionnel de son choix, et la mémoriser dans un dictionnaire de paires (réponse, mot de passe). Un tel pré calcul peut souvent être effectué en parallèle sur de nombreuses machines. Il peut alors utiliser l'attaque de texte en clair choisi pour acquérir une réponse correspondant à cette mise en cause, et simplement chercher le mot de passe dans le dictionnaire. Même si la plupart des mots de passe ne sont pas dans le dictionnaire, certains peuvent l'être. Comme l'attaquant doit répondre à la mise en cause, le coût du calcul de la

réponse pour chaque mot de passe sur la liste peut être amorti en trouvant beaucoup de mots de passe. Un dictionnaire de 100 million de paires de mots de passe/réponses prendrait environ 3,2 gigaoctets de mémoire.

La contre mesure contre cette attaque est que les clients soient configurés pour demander l'utilisation de la directive facultative "cnonce".

4.11 Attaque en force en temps différé

Avec l'authentification Digest, un MITM peut exécuter une attaque par texte en clair choisi, et peut collecter les réponses de nombreux utilisateurs au même nom occasionnel, puis trouver les mots de passe dans tout sous ensemble de l'espace de mot de passe que génèrerait une des paires nom occasionnel/réponse en un seul passage sur cet espace. Elle réduit aussi le temps nécessaire pour trouver le premier mot de passe d'un facteur égal au nombre de paires de nom occasionnel/réponse collectées. Cette recherche de l'espace de mot de passe peut souvent être effectuée en parallèle sur de nombreuses machines, et même une seule machine peut faire très rapidement une recherche sur de grands sous ensembles de l'espace de mots de passe – on rapporte des recherches sur tous les mots de passe de six lettres ou moins en quelques heures.

La contre mesure contre cette attaque est que les clients soient configurés à demander l'utilisation de la directive facultative "cnonce".

4.12 Espionnage par des serveurs falsifiés

L'authentification de base est vulnérable à la contrefaçon par des serveurs déguisés. Si un utilisateur peut être conduit à croire qu'il se connecte à un hôte qui contient des informations protégées par un mot de passe qu'il connaît, alors qu'en fait il se connecte à un serveur hostile, le serveur hostile peut alors demander un mot de passe, le mémoriser pour utilisation ultérieure, et feindre une erreur. Ce type d'attaque est plus difficile avec l'authentification Digest -- mais le client doit savoir demander que l'authentification Digest soit utilisée, peut-être en utilisant certaines des techniques décrites plus haut pour contrer les attaques "par interposition". Cette fois encore, l'utilisateur peut être aidé à détecter cette attaque par une indication visuelle du mécanisme d'authentification utilisé avec des conseils appropriés pour interpréter les implications de chaque schéma.

4.13 Mémorisation des mots de passe

L'authentification par résumé exige que l'agent d'authentification (normalement, le serveur) mémorise des données dérivées du nom et du mot de passe de l'usager dans un "fichier des mots de passe" associé à un domaine donné. Normalement il devrait contenir des paires consistant en un nom d'utilisateur et H(A1), où H(A1) est la valeur résumée du nom d'utilisateur, du domaine, et du mot de passe comme décrit ci-dessus.

Les implications de ceci sur la sécurité sont que si le fichier des mots de passe est compromis, un attaquant peut alors obtenir un accès immédiat aux documents sur le serveur qui utilisent ce domaine. À la différence, disons d'un fichier de mot de passe UNIX standard, ces informations n'ont pas besoin d'être décryptées afin d'accéder aux documents dans le domaine du serveur associé à ce fichier. D'un autre côté, le déchiffrement, ou plus vraisemblablement une attaque en force brute, serait nécessaire pour obtenir le mot de passe de l'usager. C'est la raison pour laquelle le domaine fait partie des données résumées dans le fichier de mots de passe. Cela signifie que si un fichier de mots de passe d'authentification Digest est compromis, il ne compromet pas automatiquement les autres qui ont le même nom d'utilisateur et mot de passe (bien qu'ils les expose à des attaques en force brute).

Il y a deux importantes conséquences de cela pour la sécurité. Tout d'abord, le fichier des mots de passe doit être protégé comme s'il contenait des mots de passe non chiffrés, parce que pour les besoins de l'accès aux documents dans son domaine, c'est effectivement ce qu'il fait.

Une seconde conséquence en est que la chaîne du domaine doit être unique parmi tous les domaines que chaque utilisateur va vraisemblablement utiliser. En particulier une chaîne de domaine devrait inclure le nom de l'hôte qui fait l'authentification. L'incapacité du client à authentifier le serveur est une faiblesse de l'authentification Digest.

4.14 Résumé

Selon les critères de la cryptographie moderne, l'authentification Digest est faible. Mais pour une large gamme

d'objets, elle est un remplacement valable de l'authentification Basic. Elle remédie à certaines des faiblesses, mais pas à toutes, de l'authentification de base. Sa force peut varier selon les mises en œuvre. En particulier, la structure du nom occasionnel (qui dépend de la mise en œuvre de serveur) peut affecter la facilité de monter une attaque en répétition. Une gamme d'options de serveurs est appropriée dans la mesure où, par exemple, certaines mises en œuvre peuvent vouloir accepter la prise en charge par l'enveloppe de serveur de noms occasionnels à utilisation unique ou de résumés pour éliminer la possibilité de répétition. D'autres peuvent se satisfaire d'un nom occasionnel tel que celui recommandé plus haut, se restreignant à une seule adresse IP et un seul ETag ou avec une durée de vie limitée.

Le niveau plancher est que *toute* mise en œuvre conforme sera relativement faible selon les critères cryptographiques, mais *toute* mise en œuvre conforme sera très supérieure à celle de l'authentification de base.

5 Échantillon de mise en œuvre

Le code ci-après met en œuvre les calculs de H(A1), H(A2), request-digest et response-digest, et d'un programme d'essai qui calcule les valeurs utilisées dans l'exemple du paragraphe 3.5. Il utilise la mise en œuvre de MD5 tirée de la RFC 1321.

File "digcalc.h":

```
#define HASHLEN 16
typedef char HASH[HASHLEN];
#define HASHHEXLEN 32
typedef char HASHHEX[HASHHEXLEN+1];
#define IN
#define OUT

/* calcul de H(A1) selon la spécification de Digest HTTP */
void DigestCalcHA1(
   IN char * pszAlg,
   IN char * pszUserName,
   IN char * pszRealm,
   IN char * pszPassword,
   IN char * pszNonce,
   IN char * pszCNonce,
   OUT HASHHEX SessionKey
   );

/* calcul de request-digest/response-digest selon la spécification de Digest HTTP */
void DigestCalcResponse(
        IN HASHHEX HA1,           /* H(A1) */
        IN char * pszNonce,       /* nom occasionnel provenant du serveur */
        IN char * pszNonceCount,  /* 8 chiffres hexadécimaux */
        IN char * pszCNonce,      /* nom occasionnel du client */
        IN char * pszQop,         /* qop-value: "", "auth", "auth-int" */
        IN char * pszMethod,      /* méthode tirée de la demande */
        IN char * pszDigestUri,   /* URL demandé */
        IN HASHHEX HEntity,       /* H(corps d'entité) si qop="auth-int" */
        OUT HASHHEX Response      /* request-digest ou response-digest */
        );
```

File "digcalc.c":

```
#include <global.h>
#include <md5.h>
#include <string.h>
#include "digcalc.h"

void CvtHex(
   IN HASH Bin,
```

```
    OUT HASHHEX Hex
    )
{
    unsigned short i;
    unsigned char j;
    for (i = 0; i < HASHLEN; i++) {
        j = (Bin[i] >> 4) & 0xf;
        if (j <= 9)
            Hex[i*2] = (j + '0');
         else          Hex[i*2] = (j + 'a' - 10);
        j = Bin[i] & 0xf;
        if (j <= 9)
            Hex[i*2+1] = (j + '0');
         else
            Hex[i*2+1] = (j + 'a' - 10);
    };
    Hex[HASHHEXLEN] = '\0';
};

/* calculer H(A1) selon la spécification */
void DigestCalcHA1(
    IN char * pszAlg,
    IN char * pszUserName,
    IN char * pszRealm,
    IN char * pszPassword,
    IN char * pszNonce,
    IN char * pszCNonce,
    OUT HASHHEX SessionKey
    )
{
     MD5_CTX Md5Ctx;
     HASH HA1;
     HASHHEX HA1Hex;

     MD5Init(&Md5Ctx);
     MD5Update(&Md5Ctx, pszUserName, strlen(pszUserName));
     MD5Update(&Md5Ctx, ":", 1);
     MD5Update(&Md5Ctx, pszRealm, strlen(pszRealm));
     MD5Update(&Md5Ctx, ":", 1);
     MD5Update(&Md5Ctx, pszPassword, strlen(pszPassword));
     MD5Final(HA1, &Md5Ctx);
     if (stricmp(pszAlg, "md5-sess") == 0) {
          MD5Init(&Md5Ctx);
          MD5Update(&Md5Ctx, HA1, HASHLEN);
          MD5Update(&Md5Ctx, ":", 1);
          MD5Update(&Md5Ctx, pszNonce, strlen(pszNonce));
          MD5Update(&Md5Ctx, ":", 1);
          MD5Update(&Md5Ctx, pszCNonce, strlen(pszCNonce));
          MD5Final(HA1, &Md5Ctx);
     };
     CvtHex(HA1, SessionKey);
};

/* calcul de request-digest/response-digest selon la spécification de Digest HTTP */
void DigestCalcResponse(
          IN HASHHEX HA1,           /* H(A1) */
          IN char * pszNonce,       /* nom occasionnel fourni par le serveur */
          IN char * pszNonceCount   /* 8 chiffres hexadécimaux */
          IN char * pszCNonce,      /* nom occasionnel du client */
          IN char * pszQop,         /* valeur de qop : "", "auth", "auth-int" */
```

```
        IN char * pszMethod,        /* méthode tirée de la demande */
        IN char * pszDigestUri,     /* URL demandée */
        IN HASHHEX HEntity,         /* H(corps d'entité) si qop="auth-int" */
        OUT HASHHEX Response        /* request-digest ou response-digest */
        )
{
    MD5_CTX Md5Ctx;
    HASH HA2;
    HASH RespHash;
     HASHHEX HA2Hex;

// calculer H(A2)
    MD5Init(&Md5Ctx);
    MD5Update(&Md5Ctx, pszMethod, strlen(pszMethod));
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszDigestUri, strlen(pszDigestUri));
    if (stricmp(pszQop, "auth-int") == 0) {
        MD5Update(&Md5Ctx, ":", 1);
        MD5Update(&Md5Ctx, HEntity, HASHHEXLEN);
    };
    MD5Final(HA2, &Md5Ctx);
     CvtHex(HA2, HA2Hex);

// calculer la réponse
    MD5Init(&Md5Ctx);
    MD5Update(&Md5Ctx, HA1, HASHHEXLEN);
    MD5Update(&Md5Ctx, ":", 1);
    MD5Update(&Md5Ctx, pszNonce, strlen(pszNonce));
    MD5Update(&Md5Ctx, ":", 1);
    if (*pszQop) {
       MD5Update(&Md5Ctx, pszNonceCount, strlen(pszNonceCount));
       MD5Update(&Md5Ctx, ":", 1);
       MD5Update(&Md5Ctx, pszCNonce, strlen(pszCNonce));
       MD5Update(&Md5Ctx, ":", 1);
       MD5Update(&Md5Ctx, pszQop, strlen(pszQop));
       MD5Update(&Md5Ctx, ":", 1);
    };
    MD5Update(&Md5Ctx, HA2Hex, HASHHEXLEN);
    MD5Final(RespHash, &Md5Ctx);
    CvtHex(RespHash, Response);
};
```

File "digtest.c":

```
#inclut <stdio.h>
#inclut "digcalc.h"

void main(int argc, char ** argv) {

    char * pszNonce = "dcd98b7102dd2f0e8b11d0f600bfb0c093";
    char * pszCNonce = "0a4f113b";
    char * pszUser = "Mufasa";
    char * pszRealm = "testrealm@host.com";
    char * pszPass = "Circle Of Life";
    char * pszAlg = "md5";
    char szNonceCount[9] = "00000001";
    char * pszMethod = "GET";
    char * pszQop = "auth";
    char * pszURI = "/dir/index.html";
    HASHHEX HA1;
```

```
    HASHHEX HA2 = "";
    HASHHEX Response;

    DigestCalcHA1(pszAlg, pszUser, pszRealm, pszPass, pszNonce, pszCNonce, HA1);
    DigestCalcResponse(HA1, pszNonce, szNonceCount, pszCNonce, pszQop, pszMethod, pszURI, HA2, Response);
    printf("Response = %s\n", Response);
};
```

6 Remerciements

Eric W. Sink, de AbiSource, Inc., a été un des auteurs d'origine du document avant que la spécification ne subisse des révisions substantielles.

En plus des auteurs, des idées précieuses pour la création du présent document ont été fournies par Peter J. Churchyard, Ned Freed, et David M. Kristol. Jim Gettys et Larry Masinter ont édité la mise à jour du document.

7 Références

[MD5] Kaliski, B.,Robshaw, M., "Authentification de message avec MD5", CryptoBytes, Sping 1995, RSA Inc, (http://www.rsa.com/rsalabs/pubs/cryptobytes/spring95/md5.htm)

[RFC1945] T. Berners-Lee, R. Fielding, H. Frystyk, "Protocole de transfert Hypertext -- HTTP/1.0", mai 1996. *(Information)*

[RFC2616] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", juin 1999. *(D.S., MàJ par 2817)*

[RFC1321] R. Rivest, "Algorithme de résumé de message MD5", avril 1992. *(Information)*

[RFC2045] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 1 : Format des corps de message Internet", novembre 1996. *(D. S., MàJ par 2184, 2231, 5335.)*

[RFC2046] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 2 : Types de support", novembre 1996. *(D. S., MàJ par 2646, 3798, 5147.)*

[RFC2069] J. Franks et autres, "Extension à HTTP : authentification d'accès par résumé", janvier 1997. (*Obs., voir* RFC2617) *(P.S.)*

[RFC2396] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiants de ressource uniformes (URI) : Syntaxe générique", août 1998. (*Obsolète, voir* RFC3986)

[RFC2195] J. Klensin et autres, "Extension IMAP/POP AUTHorize pour mise au défi/réponse simple", septembre 1997. *(P.S.)*

[RFC2829] M. Wahl et autres, "Méthodes d'authentification pour LDAP", mai 2000. (*Obsolète, voir* RFC4513, RFC4510) *(P.S.)*

8 Adresse des auteurs

John Franks
Professor of Mathematics
Department of Mathematics
Northwestern University
Evanston, IL 60208-2730,
USA
mél : john@math.nwu.edu

Phillip M. Hallam-Baker
Principal Consultant
Verisign Inc.
301 Edgewater Place
Suite 210
Wakefield MA 01880, USA
mél : pbaker@verisign.com

Jeffery L. Hostetler
Software Craftsman
AbiSource, Inc.
6 Dunlap Court
Savoy, IL 61874
USA
mél : jeff@AbiSource.com

Scott D. Lawrence
Agranat Systems, Inc.
5 Clocktower Place, Suite 400
Maynard, MA 01754, USA
mél : lawrence@agranat.com

Paul J. Leach
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052, USA
mél : paulle@microsoft.com

Ari Luotonen
Member of Technical Staff
Netscape Communications Corporation
501 East Middlefield Road
Mountain View, CA 94043, USA

Lawrence C. Stewart
Open Market, Inc.
215 First Street
Cambridge, MA 02142, USA
mél : stewart@OpenMarket.com

9 Déclaration de droits de reproduction

Remerciement

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