Groupe de travail Réseau
Request for Comments : 2965
RFC rendue obsolète : 2109
Catégorie : En cours de normalisation

D. Kristol, Bell Laboratories, Lucent Technologies
L. Montulli, Epinions.com, Inc.
octobre 2000
Traduction Claude Brière de L'Isle

Mécanisme de gestion de l'état HTTP

Statut du présent Mémo
La présente RFC spécifie un protocole de normalisation pour la communauté Internet et appelle à des discussions et suggestions pour son amélioration. Prière de se reporter à l'édition en cours des "Internet Official Protocol Standards" (*normes officielles du protocole Internet*) (STD 1) pour connaître l'état de la normalisation et le statut du présent protocole. La distribution du présent mémo n'est soumise à aucune restriction.

Note de l'IESG
L'IESG note que ce mécanisme utilise en interne le domaine de niveau supérieur (TLD, *top-level domain*) .local lors du traitement de noms d'hôtes qui ne contiennent aucun point, et que ce mécanisme pourrait ne pas fonctionner de la façon espérée si un TLD .local réel était jamais enregistré.

Résumé
Le présent document spécifie un moyen pour créer une session à états pleins avec des demandes et réponses du protocole de transfert Hypertexte (HTTP, *Hypertext Transfer Protocol*). Il décrit trois nouveaux en-têtes, Cookie, Cookie2, et Set-Cookie2, qui portent des informations d'état entre les serveurs d'origine et les agents d'utilisateur participants. La méthode décrite ici diffère de la proposition Cookie de Netscape [Netscape], mais elle peut interopérer avec les agents d'utilisateur HTTP/1.0 qui utilisent la méthode de Netscape. (Voir la section Historique.)

Le présent document reflète l'expérience de la mise en œuvre de la RFC2109 et la rend obsolète.

1. Terminologie

Les termes agent d'utilisateur, client, serveur, mandataire, serveur d'origine, et http_URL ont la même signification que dans la spécification HTTP/1.1 [RFC2616]. Les termes abs_path et URI absolu ont la même signification que dans la spécification de la syntaxe d'URI [RFC2396].

Nom d'hôte (HN, *Host Name*) signifie le nom de domaine de l'hôte (HDN, *Host Domain Name*) ou l'adresse numérique du protocole Internet (IP) d'un hôte. Le nom de domaine pleinement qualifié est préféré ; l'utilisation des adresses IP numériques est fortement déconseillée.

Le termes "hôte demandeur" *(request-host)* et "URI demandeur" *(request-URI)* se réfèrent aux valeurs que le client enverra au serveur, respectivement comme les portions hôte (mais pas accès) et abs_path de l'URI absolu (http_URL) de la ligne du demandeur HTTP. Noter que l'hôte demandeur est un HN.

Le terme "nom d'hôte effectif" se rapporte au nom de l'hôte. Si un nom d'hôte ne contient pas de point, le nom d'hôte effectif est le nom avec la chaîne ".local" qui lui est ajoutée. Autrement, le nom d'hôte effectif est le même que le nom d'hôte. Noter que tous les noms d'hôte effectifs contiennent au moins un point.

Le terme "accès de demande" se réfère à la portion accès de l'URI absolu (http_URL) de la ligne de demande HTTP. Si l'URI absolu n'a pas d'accès explicite, l'accès de demande est le HTTP par défaut, 80. L'accès de demande d'un mouchard *(cookie)* est l'accès de demande de la demande dans laquelle un en-tête de réponse Set-Cookie2 a été retourné à l'agent d'utilisateur.

Les noms d'hôte peuvent être spécifiés soit comme une adresse IP, soit comme une chaîne HDN. On compare parfois un nom d'hôte avec un autre. (De telles comparaisons DEVRONT être insensibles à la casse.) Le nom de domaine de l'hôte A correspond à celui de l'hôte B si :

* la comparaison de leurs chaîne de nom d'hôte donne l'égalité ;
* A est une chaîne HDN et a la forme "NB", où N est une chaîne de nom non vide, B a la forme ".B'", et B' est une chaîne HDN. (Ainsi, le domaine "x.y.com" correspond à ".Y.com" mais pas "Y.com".)

Noter que la correspondance de domaine n'est pas une opération commutative : le domaine "a.b.c.com" correspond à ."c.com", mais pas l'inverse.

La portée R d'un nom d'hôte H est définie comme suit :

* Si
  - H est le nom de domaine hôte d'un hôte ; et
  - H a la forme A.B ; et
  - A n'a pas de point incorporé (c'est-à-dire, intérieur) ; et
  - B a au moins un point incorporé, ou B est la chaîne "local",

  alors la portée de H est .B.
* Autrement, la portée de H est H.

Pour deux chaînes qui représentent des chemins, P1 et P2, le chemin P1 correspond à P2 si P2 est un préfixe de P1 (y compris le cas où la comparaison des chaînes P1 et P2 donne l'égalité). Donc, la chaîne /tec/waldo correspond au chemin /tec.

Parce qu'il était utilisé dans la mise en œuvre d'origine de la gestion d'état de Netscape, nous utiliserons le terme de mouchard *(cookie)* pour nous référer aux informations d'état qui passent entre un serveur d'origine et un agent d'utilisateur, et qui sont mémorisées par l'agent d'utilisateur.

1.1 Exigences

Les mots clés "PEUT", "DOIT", "NE DOIT PAS", "FACULTATIF", "RECOMMANDE", "EXIGE", "DEVRA", NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS" dans ce document sont à interpréter comme décrit dans la [RFC2119].

2. État et sessions

Le présent document décrit un moyen pour créer des sessions à états pleins avec les demandes et réponses HTTP. Actuellement, les serveurs HTTP répondent à chaque demande de client sans mettre en rapport cette demande avec les demandes précédentes ou suivantes ; le mécanisme de gestion d'état permet aux clients et serveurs qui souhaitent échanger des informations d'état de placer des demandes et réponses HTTP au sein d'un contexte plus large, qu'on appelle une "session". Ce contexte peut être utilisé pour créer, par exemple, un "panier " dans lequel les choix de l'utilisateur peuvent être agrégés avant l'achat, ou un système de feuilletage de magazine, dans lequel les lectures précédentes d'un usager affectent les offres qui sont présentées.

Ni les clients ni les serveurs ne sont obligés d'accepter les mouchards. Un serveur PEUT refuser de fournir un contenu à un client qui ne retourne pas les mouchards qu'il avait envoyé.

3. Description

On décrit ici un moyen pour qu'un serveur d'origine envoie des informations d'état à l'agent d'utilisateur, et pour que l'agent d'utilisateur retourne les informations d'état au serveur d'origine. Le but est d'avoir un impact minimal sur HTTP et les agents d'utilisateur.

3.1 Syntaxe : généralités

Les deux en-têtes de gestion d'état, Set-Cookie2 et Cookie, ont des propriétés syntaxiques communes qui impliquent des paires de valeur et d'attribut. La grammaire suivante utilise la notation, et les jetons DIGIT (chiffres décimaux), jeton (informellement, une séquence de caractères non spéciaux, qui ne sont pas des espaces) et http_URL de la spécification HTTP/1.1 [RFC2616] pour décrite leur syntaxe.

```
av-pairs   =    av-pair *(";" av-pair)
av-pair    =    attr ["=" valeur]              ; valeur facultative
attr       =    jeton
valeur     =    jeton | chaîne entre guillemets
```

Les attributs (nom) (attr) sont insensibles à la casse. Les espaces sont permises entre les jetons. Noter qu'alors que la description de syntaxe ci-dessus montre que valeur est facultatif, la plupart des attributs l'exigent.

Note : La syntaxe ci-dessus permet l'espace entre l'attribut et le signe =.

3.2 Rôle de serveur d'origine

3.2.1 Généralités

Le serveur d'origine initie une session, si il le désire. Pour ce faire, il retourne un en-tête supplémentaire de réponse au client, Set-Cookie2. (Les détails figurent plus loin.)

Un agent d'utilisateur retourne un en-tête de demande Cookie (voir ci-dessous) au serveur d'origine si il choisit de continuer une session. Le serveur d'origine PEUT l'ignorer ou l'utiliser pour déterminer l'état en cours de la session. Il PEUT renvoyer au client un en-tête de réponse Set-Cookie2 avec les mêmes informations ou des informations différentes, ou il PEUT n'envoyer aucun en-tête Set-Cookie2 du tout. Le serveur d'origine termine effectivement une session en envoyant au client un en-tête Set-Cookie2 avec Max-Age=0.

Les serveurs PEUVENT retourner des en-têtes de réponse Set-Cookie2 avec toute réponse. Les agents d'utilisateur DEVRAIENT envoyer des en-têtes de demande Cookie, sous réserve des autres règles précisées ci-dessous, avec chaque demande.

Un serveur d'origine PEUT inclure plusieurs en-têtes Set-Cookie2 dans une réponse. Noter qu'une passerelle intermédiaire pourrait replier de tels en-têtes en un seul en-tête.

3.2.2 Syntaxe de Set-Cookie2

La syntaxe de l'en-tête de réponse Set-Cookie2 est :

```
set-cookie     =       "Set-Cookie2:" cookies
cookies        =       1#cookie
cookie         =       NAME "=" VALUE *(";" set-cookie-av)
NAME           =       attr
VALUE          =       value
set-cookie-av  =       "Comment" "=" value
               |       "CommentURL" "=" <"> http_URL <">
               |       "Discard"
               |       "Domain" "=" value
               |       "Max-Age" "=" value
               |       "Path" "=" value
               |       "Port" [ "=" <"> portlist <"> ]
               |       "Secure"
               |       "Version" "=" 1*DIGIT
portlist       =       1#portnum
portnum        =       1*DIGIT
```

De façon informelle, l'en-tête de réponse Set-Cookie2 comporte le jeton Set-Cookie2:, suivi par une liste séparée par des virgules de un ou plusieurs mouchards. Chaque mouchard commence par une paire NOM=VALEUR, suivie par zéro, une ou plusieurs paires attribut-valeur séparées par des points-virgules. La syntaxe des paires attribut-valeur a été présentée plus haut. Les attributs spécifiques et la sémantique de leurs valeurs suivent. La paire d'attribut-valeur NOM=VALEUR DOIT venir en premier dans chaque mouchard. Les autres, s'il en est, peuvent arriver dans n'importe quel ordre. Si un attribut apparaît plus d'une fois dans un mouchard, le client DEVRA utiliser seulement la valeur associée à la première apparition de l'attribut ; un client DOIT ignorer les valeurs après la première.

Le NOM d'un mouchard PEUT être le même que celui des attributs dans sa spécification. Cependant, comme le NOM du mouchard doit venir en premier dans un en-tête de réponse Set-Cookie2, le NOM et sa VALEUR ne peuvent pas être confondus avec une paire attribut-valeur.

NOM=VALEUR
EXIGÉ. Le nom de l'information d'état ("cookie") est NOM, et sa valeur est VALEUR. Les NOM qui commencent pas $ sont réservés et NE DOIVENT PAS être utilisés par les applications.

La VALEUR est opaque à l'agent d'utilisateur et peut être toute chose que le serveur d'origine choisit d'envoyer, éventuellement sous un codage imprimable en ASCII choisi par le serveur. "Opaque" implique que le contenu ne présente un intérêt et une pertinence que pour le serveur d'origine. Le contenu peut, en fait, être lisible par toute personne qui

examine l'en-tête Set-Cookie2.

Comment=valeur
FACULTATIF. Comme les mouchards peuvent être utilisés pour déduire ou mémoriser des informations privées sur un utilisateur, la valeur de l'attribut Comment permet à un serveur d'origine de documenter comment il entend utiliser le mouchard. L'usager peut inspecter les informations pour décider d'initier ou continuer une session avec ce mouchard. Les caractères dans la valeur DOIVENT être codés en UTF-8 [RFC2279].

CommentURL="http_URL"
FACULTATIF. Comme les mouchards peuvent être utilisés pour déduire ou mémoriser des informations privées sur un utilisateur, l'attribut CommentURL permet à un serveur d'origine de documenter comment il entend utiliser le mouchard. L'usager peut inspecter les informations identifiées par l'URL pour décider d'initier ou continuer une session avec ce mouchard.

Discard
FACULTATIF. L'attribut Discard donne à l'agent d'utilisateur l'ordre d'éliminer sans condition le mouchard lorsque l'agent d'utilisateur se termine.

Domaine=valeur
FACULTATIF. La valeur de l'attribut Domaine spécifie le domaine pour lequel le mouchard est valide. Si une valeur explicitement spécifiée ne commence pas par un point, l'agent d'utilisateur met un point devant.

Max-Age=valeur
FACULTATIF. La valeur de l'attribut Max-Age est delta secondes, la durée de vie du mouchard en secondes, un entier décimal non négatif. Pour traiter correctement les mouchards placés en antémémoire, un client DEVRAIT calculer l'âge du mouchard conformément aux règles de calcul de l'âge dans la spécification HTTP/1.1 [RFC2616]. Lorsque l'âge est supérieur à delta secondes, le client DEVRAIT éliminer le mouchard. Une valeur de zéro signifie que le mouchard DEVRAIT être éliminé immédiatement.

Path=valeur
FACULTATIF. La valeur de l'attribut Path spécifie le sous ensemble des URL sur le serveur d'origine auxquels le mouchard s'applique.

Port["portlist"]
FACULTATIF. L'attribut Port restreint l'accès auquel un mouchard peut être retourné dans un en-tête de demande Cookie. Noter que la syntaxe EXIGE des guillemets autour d'une liste d'accès facultative même si il n'y a qu'un nom d'accès "portnum" dans la liste d'acccès "portlist".

Secure
FACULTATIF. L'attribut Secure (sans valeur) ordonne à l'agent d'utilisateur de n'utiliser que des moyens sûrs (non spécifiés) pour contacter le serveur d'origine chaque fois qu'il renvoie ce mouchard, pour protéger la confidentialité et l'authenticité des informations portées dans le mouchard.

L'agent d'utilisateur (éventuellement en interaction avec l'usager) PEUT déterminer quel niveau de sécurité il considère approprié pour les mouchards "sûrs". L'attribut Secure devrait être considéré comme un conseil de sécurité du serveur à l'agent d'utilisateur, indiquant qu'il est dans l'intérêt de la session de protéger le contenu du mouchard. Lorsque il renvoie un mouchard "sûr" à un serveur, l'agent d'utilisateur DEVRAIT utiliser un niveau de sécurité non inférieur à celui qu'avait le mouchard reçu de serveur.

Version=valeur
EXIGÉ. La valeur de l'attribut Version, un entier décimal, identifie la version de la spécification de gestion d'état à laquelle se conforme le mouchard. Pour la présente spécification, on applique Version=1.

3.2.3 Contrôler la mise en antémémoire

Un serveur d'origine doit être conscient de l'effet de l'éventuelle mise en antémémoire aussi bien de la ressource retournée que de l'en-tête Set-Cookie2. La mise en antémémoire de documents "publics" est désirable. Par exemple, si le serveur d'origine veut utiliser un document public tel qu'une page "d'accueil" comme sentinelle pour indiquer le début d'une session pour laquelle un en-tête de réponse Set-Cookie2 doit être généré, la page DEVRAIT être mémorisée dans des antémémoires "pré-expirées" de telle sorte que le serveur d'origine puisse voir les demandes ultérieures. Les "documents privés", par exemple ceux qui contiennent des informations strictement privées d'une session, NE DEVRAIENT PAS être mise en antémémoire dans des antémémoires partagées.

Si le mouchard est destiné à être utilisé par un seul usager, l'en-tête Set-Cookie2 NE DEVRAIT PAS être mis en antémémoire. Un en-tête Set-Cookie2 qui est destiné à être partagé par plusieurs usagers PEUT être mis en antémémoire.

Le serveur d'origine DEVRAIT envoyer les en-têtes de réponse HTTP/1.1 supplémentaires, selon les circonstances :

* Pour supprimer la mise en antémémoire de l'en-tête Set-Cookie2 : Cache-control: no-cache="set-cookie2"

et un des suivants :

* Pour supprimer la mise en antémémoire d'un document privé dans des antémémoires partagées : Cache-control: private

* Pour permettre la mise en antémémoire d'un document et exiger qu'il soit validé avant de le retourner au client :

  Cache-Control: must-revalidate, max-age=0

* Pour permettre la mise en antémémoire d'un document, mais exiger que les antémémoires mandataires (pas les antémémoires d'agent d'utilisateur) le valident avant de le retourner au client :

  Cache-Control: proxy-revalidate, max-age=0

* Pour permettre la mise en antémémoire d'un document et demander qu'il soit validé avant de le retourner au client (en le "pré-expirant") : Cache-control: max-age=0
  Toutes les antémémoires ne vont pas revalider le document dans tous les cas.

Les serveurs HTTP/1.1 DOIVENT envoyer Expires: old-date (où old-date est une date loin dans le passé) sur les réponses qui contiennent des en-têtes de réponse Set-Cookie2 sauf si ils tiennent pour certain (par des moyens hors bande) qu'il n'y a pas de mandataire HTTP/1.0 dans la chaîne de réponse. Les serveurs HTTP/1.1 PEUVENT envoyer d'autres directives Cache-Control qui permettent la mise en antémémoire par les mandataires HTTP/1.1 en plus de la directive Expires: old-date ; la directive Cache-Control va outrepasser la directive Expires: old-date pour les mandataires HTTP/1.1.

3.3 Rôle d'agent d'utilisateur

3.3.1 Interpréter Set-Cookie2

L'agent d'utilisateur garde trace séparément des informations d'état qui arrivent via les en-têtes de réponse Set-Cookie2 de chaque serveur d'origine (distingués par le nom ou l'adresse et l'accès IP). L'agent d'utilisateur DOIT ignorer les paires attribut-valeur dont il ne reconnaît pas l'attribut. L'agent d'utilisateur appliques les valeurs par défaut suivantes pour les attributs facultatifs qui manquent :

Discard
Le comportement par défaut est dicté par la présence ou l'absence d'un attribut Max-Age.

Domain
On revient par défaut à l'hôte demandeur effectif. (Noter que parce qu'il n'y a pas de point au début de l'hôte demandeur effectif, le domaine par défaut ne peut confronter son domaine qu'à lui-même.)

Max-Age
Le comportement par défaut est d'éliminer le mouchard lorsque l'agent d'utilisateur quitte.

Path
On revient par défaut au chemin de l'URL de demande qui a généré la réponse Set-Cookie2, jusque et y inclus la barre oblique inversée "/" la plus à droite.

Port
Le comportement par défaut pour l'accès est qu'un mouchard PEUT être retourné à n'importe quel accès de la demande.

Secure
Si il est absent, l'agent d'utilisateur PEUT envoyer le mouchard sur un canal non sécurisé.

3.3.2 Rejet des mouchards

Pour empêcher d'éventuelles violations de la sécurité ou de la confidentialité, un agent d'utilisateur rejette un mouchard conformément aux règles ci-dessous. Le but de ces règles est d'essayer de limiter l'ensemble des serveurs pour lesquels un mouchard est valide, sur la base des valeurs des attributs Path, Domain, et Port et de l'URI, de l'hôte, et de l'accès de la demande.

Un agent d'utilisateur rejette (il NE DEVRA PAS mémoriser ses informations) si l'attribut Version manque. De plus, un agent d'utilisateur rejette (il NE DEVRA PAS mémoriser ses informations) si une des conditions suivantes est vraie des attributs explicitement présents dans l'en-tête de réponse Set-Cookie2 :

* La valeur pour l'attribut Path n'est pas un préfixe de l'URI de la demande.
* La valeur pour l'attribut Domain ne contient pas de point incorporé, et la valeur n'est pas ".local".
* Le nom d'hôte effectif qui se déduit de l'hôte de la demande ne correspond pas à l'attribut Domain.
* L'hôte de la demande est un HDN (et non une adresse IP) et a la forme HD, où D est la valeur de l'attribut Domain, et H est une chaîne qui contient un ou plusieurs points.
* L'attribut Port a une "port-list", et l'accès de la demande ne figure pas dans la liste.

Exemples :

* Un Set-Cookie2 provenant de l'hôte de demande y.x.foo.com pour Domain=.foo.com sera rejeté parce que H est y.x et contient un point.
* Un Set-Cookie2 provenant de l'hôte de demande x.foo.com pour Domain=.foo.com serait accepté.
* Un Set-Cookie2 avec Domain=.com ou Domain=.com., sera toujours rejeté, parce que il n'y a pas de point incorporé.
* Un Set-Cookie2 avec Domain=ajax.com sera accepté, et la valeur pour Domain sera prise comme étant .ajax.com, parce que un point est ajouté devant la valeur.
* Un Set-Cookie2 avec Port="80,8000" sera accepté si la demande était faite à l'accès 80 ou 8000 et sera rejeté autrement.
* Un Set-Cookie2 provenant de "request-host example" pour Domain=.local sera accepté, parce que le nom d'hôte effectif pour l'hôte de la demande est example.local, et example.local est un domaine qui correspond à ".local".

3.3.3 Gestion du mouchard

Si un agent d'utilisateur reçoit un en-tête de réponse Set-Cookie2 dont le NOM est le même que celui d'un mouchard qu'il a mémorisé précédemment, le nouveau mouchard supplante l'ancien lorsque les valeurs de l'attribut Domain de l'ancien et du nouveau sont égales par comparaison, en utilisant une comparaison de chaîne insensible à la casse, et que les valeurs de l'attribut Path ancien et nouveau sont égales par comparaison de chaîne (sensible à la casse). Cependant, si le Set-Cookie2 a une valeur de Max-Age de zéro, le mouchard (ancien et nouveau) est éliminé. Autrement, un mouchard persiste (si les ressources le permettent) jusqu'à selon ce qui arrive d'abord avant d'être éliminé : sa durée de vie Max-Age est épuisée, ou, si l'attribut Discard est mis, que l'agent d'utilisateur termine la session.

Parce que les agents d'utilisateur ont un espace fini pour mémoriser les mouchards, ils PEUVENT aussi éliminer les plus anciens mouchards pour faire de la place pour les nouveaux, en utilisant, par exemple, un algorithme du moins récemment utilisé, ainsi que des contraintes sur le nombre maximum de mouchards que chaque serveur d'origine peut établir.

Si un en-tête de réponse Set-Cookie2 comporte un attribut Comment, l'agent d'utilisateur DEVRAIT mémoriser cette information sous une forme lisible par l'homme avec le mouchard, et DEVRAIT afficher le texte du commentaire au titre de l'inspection des mouchards à l'interface d'utilisateur.

Si un en-tête de réponse Set-Cookie2 comporte un attribut CommentURL, l'agent d'utilisateur DEVRAIT mémoriser cette information sous une forme lisible par l'homme avec le mouchard, ou, de préférence, DEVRAIT permettre à l'usager de suivre le lien http_URL au titre de l'inspection des mouchards à l'interface d'utilisateur.

L'inspection des mouchards à l'interface d'utilisateur peut comporter une facilité par laquelle un usager peut décider, au moment où l'agent d'utilisateur reçoit l'en-tête de réponse Set-Cookie2, si il accepte ou non le mouchard. Une situation potentiellement confuse pourrait survenir si la séquence suivante se présente :

* l'agent d'utilisateur reçoit un mouchard qui contient un attribut CommentURL ;
* L'interface d'inspection des mouchards de l'agent d'utilisateur est configurée de telle sorte qu'elle présente un dialogue à l'usager avant que l'agent d'utilisateur accepte le mouchard ;
* le dialogue permet à l'usager de suivre le lien CommentURL lorsque l'agent d'utilisateur reçoit le mouchard ; et
* lorsque l'usager suit le lien CommentURL, le serveur d'origine (ou un autre serveur, via d'autres liens dans le contenu retourné) retourne un autre mouchard.

L'agent d'utilisateur NE DEVRAIT PAS envoyer de mouchard dans ce contexte. L'agent d'utilisateur PEUT éliminer tout mouchard qu'il reçoit dans ce contexte si l'usager ne l'a pas, par un mécanisme de l'agent d'utilisateur, trouvé acceptable.

Les agents d'utilisateur DEVRAIENT permettre à l'usager de contrôler la destruction des mouchards, mais ils NE DOIVENT PAS étendre la durée de vie des mouchards au delà de ce qui est contrôlé par les attributs Discard et Max-Age. Un mouchard peu fréquemment utilisé peut fonctionner comme un "fichier de préférences" pour des applications réseau, et un usager peut souhaiter le garder même si c'est le mouchard le moins récemment utilisé. Une mise en œuvre possible serait qu'une interface permette la mémorisation permanente d'un mouchard au moyen d'un boîtier de vérification (ou, à l'inverse, sa destruction immédiate).

Les considérations de confidentialité incitent à ce que l'usager ait un contrôle considérable sur la gestion des mouchards. La section Confidentialité contient d'autres informations.

3.3.4 Envoi des mouchards au serveur d'origine

Lorsque il envoie une demande à un serveur d'origine, l'agent d'utilisateur inclut un en-tête de demande Cookie si il a mémorisé des mouchards qui sont applicables à la demande, sur la base de :

* l'hôte de demande et l'accès de demande ;
* l'URI de demande ;
* l'âge du mouchard.

La syntaxe de l'en-tête est :

```
cookie          =  "Cookie:" cookie-version 1*((";" | ",") cookie-valeur)
cookie-valeur   =  NOM "=" VALEUR [";" chemin] [";" domaine] [";" accès]
cookie-version  =  "$Version" "=" valeur
NOM             =  attr
VALEUR          =  valeur
chemin          =  "$Path" "=" valeur
domaine         =  "$Domain" "=" valeur
accès           =  "$Port" [ "=" <"> valeur <"> ]
```

La valeur de l'attribut cookie-version DOIT être la valeur venant de l'attribut Version de l'en-tête de réponse Set-Cookie2 correspondant. Autrement la valeur pour cookie-version est 0. La valeur pour l'attribut chemin DOIT être la valeur venant de l'attribut Path, si il en était un présent, de l'en-tête de réponse Set-Cookie2 correspondant. Autrement, l'attribut DEVRAIT être omis de l'en-tête de demande Cookie. La valeur pour l'attribut Domaine DOIT être la valeur venant de l'attribut Domaine, si il en était un présent, de l'en-tête de réponse Set-Cookie2 correspondant. Autrement, l'attribut DEVRAIT être omis de l'en-tête de demande Cookie.

L'attribut accès de l'en-tête de demande Cookie DOIT refléter l'attribut Accès, si il en était un présent, dans l'en-tête de réponse Set-Cookie2 correspondant. C'est-à-dire, l'attribut accès DOIT être présent si l'attribut Accès était présent dans l'en-tête Set-Cookie2, et il DOIT avoir la même valeur, si il y en a une. Autrement, si l'attribut Accès était absent de l'en-tête Set-Cookie2, l'attribut DOIT de même être omis de l'en-tête de demande Cookie.

Noter qu'il n'y a d'attribut ni Commentaire ni CommentURL dans l'en-tête de demande Cookie correspondant à ceux de l'en-tête de réponse Set- Cookie2. L'agent d'utilisateur ne retourne pas les informations de commentaire au serveur d'origine.

L'agent d'utilisateur applique les règles suivantes pour choisir les valeurs de mouchard applicables à envoyer dans les en-têtes de demande Cookie parmi tous les mouchards qu'il a reçu.

Selection du domaine
Le nom d'hôte effectif du serveur d'origine DOIT correspondre au domaine de l'attribut Domain du mouchard.

Selection de l'accès
Il y a trois comportements possibles, selon l'attribut Accès dans l'en-tête de réponse Set-Cookie2 :

1. par défaut (pas d'attribut Accès), le mouchard PEUT être envoyé à n'importe quel accès,
2. si l'attribut est présent mais n'a pas de valeur (par exemple, Port), le mouchard DOIT seulement être envoyé à l'accès de la demande d'où il a été reçu,
3 si l'attribut a une liste d'accès, le mouchard DOIT seulement être retourné si le nouvel accès de la demande est un de ceux de la liste des accès.

Sélection du chemin
L'URI de la demande DOIT correspondre au chemin de l'attribut Chemin du mouchard.

Sélection de Max-Age
Les mouchards qui sont arrivés à expiration devraient avoir été éliminés et donc ne sont pas transmis à un serveur d'origine.

Si plusieurs mouchards satisfont aux critères ci-dessus, ils sont rangés dans l'en-tête Cookie de telle façon que ceux qui ont les attributs Chemin les plus spécifiques précèdent ceux qui sont moins spécifiques. L'ordre par rapport aux autres attributs (par exemple, Domaine) n'est pas spécifié.

Note : Pour la rétro compatibilité, le séparateur dans l'en-tête Cookie est partout un point-virgule (;). Un serveur DEVRAIT aussi accepter la virgule (,) comme séparateur entre les valeurs de mouchards pour la compatibilité future.

3.3.5 Identifier quelle version est comprise : Cookie2

L'en-tête de demande Cookie2 facilite l'interopération entre clients et serveurs qui comprennent des versions différentes de la spécification des mouchards. Lorsque le client envoie un ou plusieurs mouchards à un serveur d'origine, si au moins un de ces mouchards contient un attribut $Version dont la valeur est différente de la version que le client comprend, le client DOIT alors aussi envoyer un en-tête de demande Cookie2, dont la syntaxe est :

cookie2 = "Cookie2:" cookie-version

Ici, la valeur pour cookie-version est la version la plus élevée de la spécification de mouchard (actuellement 1) que le client comprend. Le client doit envoyer au plus un de ces en-têtes de demande par demande.

3.3.6 Envoi de mouchards dans des transactions invérifiables

Les usagers DOIVENT avoir le contrôle des sessions afin de s'assurer de sa confidentialité. (Voir ci-dessous la section 6 "Confidentialité".) Pour simplifier la mise en œuvre et empêcher d'ajouter une couche de complexité là où existent des garde-fous adéquats, le présent document distingue cependant entre les transactions qui sont vérifiables et celles qui ne le sont pas. Une transaction est vérifiable si l'usager, ou un agent désigné par l'usager, a la faculté de revoir l'URI de la demande avant de l'utiliser dans la transaction. Une transaction est invérifiable si l'usager n'a pas cette option. Les transactions invérifiables surviennent normalement lorsque un agent d'utilisateur demande automatiquement des entités en ligne ou incorporées ou quand il résout des réponses de redirection (3xx) provenant d'un serveur d'origine. Normalement, la transaction d'origine, celle que l'usager initie, est vérifiable, et cette transaction peut induire directement ou indirectement l'agent d'utilisateur à faire des transactions invérifiables.

Une transaction invérifiable est vers un hôte tiers si le domaine de l'hôte de demande U ne correspond pas à la portée de l'hôte de demande O dans la transaction d'origine.

Lorsque il fait une transaction invérifiable, un agent d'utilisateur DOIT désactiver tout traitement de mouchard (c'est-à-dire, il NE DOIT PAS envoyer de mouchards, et NE DOIT PAS accepter de recevoir de mouchard) si la transaction est avec un hôte tiers.

Cette restriction empêche un auteur de service malveillant d'utiliser des transactions invérifiables pour conduire un agent d'utilisateur à commencer ou continuer une session avec un serveur dans un domaine différent. Le commencement ou la continuation de telles sessions pourrait être contraire aux attentes de confidentialité de l'usager, et pourrait aussi poser un problème de sécurité.

Les agents d'utilisateur PEUVENT offrir des options configurables qui lui permettent, à lui ou à tout programme autonome qu'exécute l'agent d'utilisateur, d'ignorer la règle ci-dessus, pour autant que ces options d'outrepassement soient désactivées par défaut.

(Note : On peut proposer des mécanismes qui automatisent l'outrepassement des restrictions concernant les tiers dans des conditions contrôlées.)

De nombreux agents d'utilisateur actuels fournissent déjà une option de révision qui rend vérifiables beaucoup de liens. Par exemple, certains agents d'utilisateur affichent l'URL qui serait référencé pour un lien particulier lorsque le pointeur de la souris est placé sur ce lien. L'usager peut donc déterminer si il veut visiter ce site avant de faire que le navigateur le fasse. (Bien qu'elle ne soit pas mise en œuvre sur les agents d'utilisateur actuels, une technique similaire pourrait être utilisée avec un bouton servant à proposer un formulaire – l'agent d'utilisateur pourrait afficher l'action à entreprendre si l'usager choisit ce bouton.) Cependant, même cela ne rendrait pas tous les liens vérifiables ; par exemple, les liens sur des images chargées automatiquement ne seraient normalement pas soumis à une vérification par le pointeur de la souris.

De nombreux agents d'utilisateur fournissent aussi à l'usager l'option de voir la source HTML d'un document, ou de sauvegarder la source sur un fichier externe où il peut être vu par une autre application. Bien qu'une telle option fournisse

un mécanisme brut de vérification, certains usagers peuvent ne pas le considérer comme acceptable pour cela .

3.4 Comment un serveur d'origine interprète l'en-tête Cookie

Un agent d'utilisateur retourne beaucoup des informations dans l'en-tête Set-Cookie2 au serveur d'origine lorsque le chemin de l'URI de la demande correspond à l'attribut Chemin dans le mouchard. Lorsque il reçoit un en-tête Cookie, le serveur d'origine DEVRAIT traiter les mouchards qui ont des NOM dont le préfixe est $ en particulier, comme un attribut pour le mouchard.

3.5 Rôle du mandataire d'antémémoire

Une raison de séparer les informations d'état d'un URL et d'un contenu de document est de faciliter l'adaptabilité que permet la mise en antémémoire. Pour prendre en charge les mouchards, un mandataire d'antémémoire DOIT obéir aux règles qui figurent déjà dans la spécification HTTP :

* honorer les demandes provenant de l'antémémoire, si possible, sur la base des règles de validité de l'antémémoire,
* passer un en-tête de demande Cookie dans toute demande que le mandataire doit faire à un autre serveur,
* retourner la réponse au client, et y inclure tout en-tête de réponse Set-Cookie2,
* mettre en antémémoire la réponse reçue sous réserve du contrôle des en-têtes usuels, tels que "Expires", "Cache-control: no-cache" et "Cache-control: private",
* mettre en antémémoire le Set-Cookie2 sous réserve du contrôle des en-têtes usuels, Cache-control: no-cache="set-cookie2" (l'en-tête Set-Cookie2 ne devrait normalement pas être mis en antémémoire.)

Les mandataires NE DOIVENT PAS introduire d'en-têtes Set-Cookie2 (Cookie) de leur cru dans leurs réponses (demandes) de mandataires.

4. Exemples

4.1 Exemple 1

La plupart des détails des en-têtes de demande et de réponse ont été omis. On suppose que l'agent d'utilisateur n'a pas mémorisé les mouchards.

1. Agent d'utilisateur --> Serveur

   POST /acme/login HTTP/1.1 [données du formulaire]

L'usager s'identifie via un formulaire.

2. Serveur --> Agent d'utilisateur

   HTTP/1.1 200 OK
   Set-Cookie2: Customer="WILE_E_COYOTE"; Version="1"; Path="/acme"

Le mouchard reflète l'identité de l'usager.

3. Agent d'utilisateur --> Serveur

   POST /acme/pickitem HTTP/1.1
   Cookie: $Version="1"; Customer="WILE_E_COYOTE"; $Path="/acme" [données du formulaire]

L'usager sélectionne un élément pour le "panier d'achats".

4. Serveur --> Agent d'utilisateur

   HTTP/1.1 200 OK
   Set-Cookie2: Part_Number="Rocket_Launcher_0001"; Version="1"; Path="/acme"

Le panier d'achats contient un élément.

5. Agent d'utilisateur --> Serveur

```
POST /acme/shipping HTTP/1.1
Cookie: $Version="1";
      Customer="WILE_E_COYOTE"; $Path="/acme";
      Part_Number="Rocket_Launcher_0001"; $Path="/acme" [données  du formulaire]
```

L'usager sélectionne une méthode d'expédition dans le formulaire.

6. Serveur --> Agent

```
HTTP/1.1 200 OK
Set-Cookie2: Shipping="FedEx"; Version="1"; Path="/acme"
```

Le nouveau mouchard reflète la méthode d'expédition.

7. Agent d'utilisateur --> Serveur

```
POST /acme/process HTTP/1.1
Cookie: $Version="1";
      Customer="WILE_E_COYOTE"; $Path="/acme";
      Part_Number="Rocket_Launcher_0001"; $Path="/acme";
      Shipping="FedEx"; $Path="/acme" [données  du formulaire]
```

L'usager choisit de traiter l'ordre.

8. Serveur --> Agent d'utilisateur

```
HTTP/1.1 200 OK
```

La transaction est terminée.

L'agent d'utilisateur fait une série de demandes au serveur d'origine, et reçoit un nouveau mouchard après chacune d'elles. Tous les mouchards ont le même attribut Path et le même domaine (par défaut). Parce que les chemins de tous les URI de demande correspondent à /acme, l'attribut Path de chaque mouchard, chaque demande contient tous les mouchards reçus jusque là.

4.2 Exemple 2

Cet exemple illustre les effets de l'attribut Path. Tous les détails des en-têtes de demande et de réponse ont été omis. On suppose que l'agent d'utilisateur n'a pas mémorisé de mouchard.

Imaginons que l'agent d'utilisateur a reçu, en réponse à des demandes antérieures, les en-têtes de réponse

Set-Cookie2: Part_Number="Rocket_Launcher_0001"; Version="1"; Path="/acme"

et

Set-Cookie2: Part_Number="Riding_Rocket_0023"; Version="1"; Path="/acme/ammo"

Une demande suivante de l'agent d'utilisateur au (même) serveur à des URL de la forme /acme/ammo/... inclurait l'en-tête de demande suivant :

```
Cookie: $Version="1";
      Part_Number="Riding_Rocket_0023"; $Path="/acme/ammo";
      Part_Number="Rocket_Launcher_0001"; $Path="/acme"
```

Noter que la paire NOM=VALEUR pour le mouchard avec l'attribut Path le plus spécifique, /acme/ammo, vient avant celui de l'attribut Path le moins spécifique, /acme. Noter de plus que le même nom de mouchard apparaît plus d'une fois.

Une demande suivante de l'agent d'utilisateur au (même) serveur pour un URL de la forme /acme/parts/ inclurait l'en-tête de demande suivant :

Cookie: $Version="1"; Part_Number="Rocket_Launcher_0001"; $Path="/acme"

Ici, le second attribut Path du mouchard , /acme/ammo n'est pas un préfixe de l'URL de demande, /acme/parts/, de sorte que le mouchard n'est pas transmis au serveur.

5. Considérations pour la mise en œuvre

On fournit ici des lignes directrices sur des détails vraisemblablement souhaitables pour un serveur d'origine qui met en œuvre la gestion d'état.

5.1 Contenu de Set-Cookie2

Le contenu d'un serveur d'origine devrait probablement être divisé en zones d'application disjointes, dont certaines exigent l'utilisation des informations d'état. Les zones d'application peuvent être distinguées par leurs URL de demande. L'en-tête Set-Cookie2 peut incorporer des informations sur les zones d'application en établissant l'attribut Path pour chacune d'elles.

Les informations de session peuvent évidemment être en texte en clair ou codé qui décrit l'état. Cependant, si elles grossissent trop, cela peut devenir peu maniable. Donc, une mise en œuvre pourrait choisir que les informations de session soient une clé pour une ressource à côté du serveur. Bien sûr, utiliser une base de données crée quelques problèmes que la présente spécification de gestion d'état était destinée à éviter, à savoir :

1. garder l'état réel du côté du serveur ;
2. comment et quand purger les entrées de la base de données, au cas où l'agent d'utilisateur termine la session pour, par exemple, sortir de l'application.

5.2 Pages sans état

La mise en antémémoire bénéficie de la capacité d'adaptation de la Toile mondiale. Il est donc important de réduire le nombre de documents qui ont par nature un état incorporé. Par exemple, si une application de type panier d'achats affiche toujours le contenu actuel du panier d'un usager sur chaque page, ces pages ne peuvent pas être mises en antémémoire, parce que le contenu du panier de chaque usager va être différent. D'un autre côté, si chaque page contient juste un lien qui permet à l'usager de "Voir mon panier d'achat", la page peut être mise en antémémoire.

5.3 Limites de mise en œuvre

Les mises en œuvre pratiques d'agent d'utilisateur ont des limites quant au nombre et à la taille des mouchards qu'elles peuvent mémoriser. En général, la prise en charge des mouchards d'agents d'utilisateur ne devrait pas avoir de limites fixes. Elles devraient s'efforcer de mémoriser autant de mouchards fréquemment utilisés que possible. De plus, les agents d'utilisateur d'utilisation générale DEVRAIENT fournir toutes les capacités minimum individuellement, mais pas nécessairement simultanément :

* au moins 300 mouchards,
* au moins 4096 octets par mouchard (mesuré par les caractères que comporte le mouchard non terminal dans la description de syntaxe de l'en-tête Set-Cookie2, et comme reçu dans l'en-tête Set-Cookie2),
* au moins 20 mouchards par hôte unique ou nom de domaine.

Les agents d'utilisateur créés pour des objets spécifiques ou pour des appareils à capacité limitée DEVRAIENT fournir au moins 20 mouchards de 4096 octets, pour assurer que l'usager peut interagir avec un serveur d'origine sur la base de la session.

Les informations dans un en-tête de réponse Set-Cookie2 DOIVENT être conservées dans leur totalité. Si pour une raison quelconque, il y a un espace inadéquat pour mémoriser le mouchard, il DOIT être éliminé, et non tronqué.

Les applications devraient utiliser aussi peu de mouchards que possible et d'aussi petite taille que possible, et elles devraient accepter de bonne grâce la perte d'un mouchard.

5.3.1 Attaques de déni de service

Les agents d'utilisateur PEUVENT choisir de fixer une limite supérieure au nombre de mouchards à mémoriser de la part d'un hôte ou nom de domaine donné, ou sur la taille des informations du mouchard. Autrement, un serveur malveillant pourrait tenter d'inonder un agent d'utilisateur avec de nombreux, ou de gros mouchards, sur les réponses successives, ce qui forcerait l'agent d'utilisateur a éliminer les mouchards reçus d'autres serveurs. Cependant, les minimums spécifiés ci-

dessus DEVRAIENT toujours être acceptés.

6. Confidentialité

Le consentement informé devrait guider la conception des systèmes qui utilisent les mouchards. Un usager devrait être capable de découvrir comment un site de la Toile a prévu d'utiliser les informations d'un mouchard et devrai être capable de choisir si ces politiques sont ou non acceptables. L'agent d'utilisateur et le serveur d'origine doivent toux deux aider au consentement informé.

6.1 Contrôle de l'agent d'utilisateur

Un serveur d'origine pourrait créer un en-tête Set-Cookie2 pour tracer le chemin d'un usager jusqu'au serveur. Les usagers peuvent objecter à ce comportement comme étant une accumulation intrusive d'informations même si leur identité n'est pas évidente. (L'identité peut devenir évidente, par exemple, si un usager remplit ensuite un questionnaire qui contient des informations d'identification.) La présente spécification de la gestion d'état exige donc qu'un agent d'utilisateur donne à l'usager le contrôle sur de telles intrusions éventuelles, bien que l'interface à travers laquelle l'usager va exercer ce contrôle reste non spécifiée. Cependant, les mécanismes de contrôle fournis DEVRONT au moins permettre à l'usager :

* de désactiver complètement l'envoi et la mémorisation des mouchards,
* de déterminer si une session à états peins est en cours,
* de contrôler la sauvegarde d'un mouchard sur la base de l'attribut Domaine du mouchard.

Un tel contrôle pourrait être fourni, par exemple, par des mécanismes :

* pour notifier à l'usager quand l'agent d'utilisateur est sur le point d'envoyer un mouchard au serveur d'origine, pour offrir la faculté de ne pas commencer une session,
* pour afficher une indication visuelle qu'une session à états pleins est en cours,
* pour laisser l'usager décider quels mouchards, s'il en est, devraient être sauvegardés lorsque il termine une fenêtre ou une session d'agent d'utilisateur,
* pour laisser l'usager examiner et supprimer le contenu d'un mouchard à tout moment.

Un agent d'utilisateur commence normalement l'exécution sans aucune mémorisation d'informations d'état. Il DEVRAIT être possible de configurer un agent d'utilisateur pour qu'il n'envoie jamais d'en-tête Cookie, auquel cas, il ne peut jamais garder l'état avec un serveur d'origine. (L'agent d'utilisateur va alors se comporter comme un agent qui ne sait pas comment traiter les en-têtes de réponse Set-Cookie2.)

Lorsque l'agent d'utilisateur termine l'exécution, il DEVRAIT dire à l'usager d'éliminer toutes les informations d'état. Autrement, l'agent d'utilisateur PEUT demander à l'usager si les informations d'état devraient être conservées ; la valeur par défaut devrait être "non". Si l'usager choisit de conserver les informations d'état, elles seront restaurées la prochaine fois que l'agent d'utilisateur sera activé.

Note : Les agents d'utilisateur devraient probablement être prudents dans l'utilisation des fichiers pour mémoriser à long terme les mouchards. Si un usager fait fonctionner plus d'une instance de l'agent d'utilisateur, les mouchards pourraient être entremêlés ou corrompus.

6.2 Rôle du serveur d'origine

Un serveur d'origine DEVRAIT promouvoir le consentement informé en ajoutant des informations de CommentURL ou Comment aux mouchards qu'il envoie. CommentURL est préféré parce qu'il offre l'opportunité de fournir de plus riches informations dans de nombreuses langues.

6.3 Texte en clair

Les informations ne sont pas protégées dans les en-têtes Set-Cookie2 et Cookie. Par conséquent :

1. toute information sensible qui est convoyée dedans est exposée aux intrus,
2. un intermédiaire malveillant pourrait altérer les en-têtes lorsqu'ils passent dans l'une ou l'autre direction, avec des résultats imprévisibles.

Ces faits impliquent que les informations de nature personnelle et/ou financière ne devraient être envoyées que sur un canal sécurisé. Pour des informations moins sensibles, ou quand le contenu de l'en-tête est une clé de base de données, un serveur d'origine devrait être vigilant pour empêcher qu'une mauvaise valeur de Cookie cause des défaillances.

Un agent d'utilisateur dans un environnement d'utilisateur partagé court un risque supplémentaire. En utilisant une interface

d'inspection des mouchards, l'usager B pourrait examiner le contenu des mouchards qui ont été sauvegardés lorsque l'usager A s'est servi de la machine.

7. Considérations pour la sécurité

7.1 Conception du protocole

Les restrictions sur la valeur de l'attribut Domaine, et les règles concernant les transactions invérifiables, sont destinées à réduire les façons dont les mouchards peuvent "fuir" vers le "mauvais" site. L'intention est de restreindre les mouchards à un hôte, ou un ensemble d'hôtes en relations étroites. Un hôte de demande est donc limité quant aux valeurs qu'il peut établir pour Domaine. On considère qu'il est acceptable pour les hôtes host1.foo.com et host2.foo.com de partager des mouchards, mais pas pour a.com et b.com.

De même, un serveur peut établir un chemin pour les seuls mouchards qui se rapportent à l'URI de la demande.

7.2 Falsification de mouchard

Une conception appropriée de l'application peut éviter des attaques masquées de la part de domaines proches. Considérons :

1. L'agent d'utilisateur fait une demande à victim.cracker.edu, obtient en retour le mouchard session_id="1234" et établit le domaine par défaut victim.cracker.edu.

2. L'agent d'utilisateur fait une demande à spoof.cracker.edu, obtient en retour le mouchard session-id="1111", avec Domain=".cracker.edu".

3. L'agent d'utilisateur fait à nouveau une demande à victim.cracker.edu, et passe

   Cookie: $Version="1"; session_id="1234", $Version="1"; session_id="1111"; $Domain=".cracker.edu"

Le serveur à victim.cracker.edu devrait détecter que le second mouchard n'est pas un de ceux qu'il a généré en remarquant que l'attribut Domaine n'est pas pour lui-même et l'ignorer.

7.3 Partage de mouchard imprévu

Un agent d'utilisateur DEVRAIT tout faire pour empêcher le partage des informations de session entre des hôtes qui sont dans des domaines différents. Des objets incorporés ou en ligne peuvent causer des problèmes particulièrement sévères de confidentialité si ils peuvent être utilisés pour partager des mouchards entre des hôtes disparates. Par exemple, un serveur malveillant pourrait incorporer des informations de mouchard pour l'hôte "a.com" dans un URI pour une CGI *(common gateway interface,* interface de passerelle commune) sur l'hôte "b.com". Les mises en œuvre d'agent d'utilisateur sont vivement encouragées à empêcher cette sorte d'échange chaque fois que possible.

7.4 Mouchards pour les informations de comptabilité

Bien que ce soit une pratique courante de les utiliser de cette façon, les mouchards ne sont pas conçus ou destinés à être utilisés pour détenir des informations d'authentification, comme des noms de comptes et des mots de passe. Sauf si de tels mouchards sont échangés sur un chemin chiffré, les informations de comptabilité qu'ils contiennent sont très vulnérables à la lecture et au vol.

8. Autres propositions similaires

À part la RFC2109, trois autres propositions ont été faites pour atteindre des buts similaires. La présente spécification a commencé comme un amalgame de la proposition d'informations d'état de Kristol [DMK95] et de la proposition Cookie de Netscape [Netscape].

Brian Behlendorf a proposé un en-tête Session-ID qui serait initié par l'agent d'utilisateur et pourrait être utilisé par un serveur d'origine pour traquer les "lignes de clics". Il ne porterait cependant aucun état défini par le serveur d'origine. Phillip Hallam-Baker a proposé un autre mécanisme d'identifiant de session défini par le client pour des fins similaires.

Bien qu'identifiants de session et mouchards puissent tous deux fournir un moyen pour conserver des sessions à états pleins, leur objet de départ est différent, et, par conséquent, leurs exigences de confidentialité sont différentes. Un usager initie des identifiants de session pour permettre aux serveurs de tracer la progression entre eux, ou pour distinguer plusieurs usagers sur une machine partagée. Les mouchards sont initié par le serveur, de sorte que le mécanisme de mouchard décrit ici donne aux usagers le contrôle sur quelque chose qui autrement aurait lieu à l'insu des usagers. De plus, les mouchards convoient des informations riches, choisies par le serveur, alors que les identifiants de session comportent des informations simples, choisies par l'usager.

9. Historique

9.1 Compatibilité avec les mises en œuvre existantes

Les mises en œuvre existantes de mouchards, fondées sur la spécification de Netscape, utilisent l'en-tête Set-Cookie (et non Set-Cookie2). Les agents d'utilisateur qui reçoivent dans la même réponse à la fois un en-tête de réponse Set-Cookie et Set-Cookie2 pour le même mouchard DOIVENT éliminer les informations du Set-Cookie et utiliser seulement les informations du Set-Cookie2. De plus, un agent d'utilisateur DOIT supposer, si il a reçu un en-tête de réponse Set-Cookie2, que le serveur qui l'a envoyé se conforme au présent document et va comprendre les en-têtes de demande Cookie qui suivent aussi cette spécification.

Les nouveaux mouchards DOIVENT remplacer à la fois les mouchards équivalents d'ancien et de nouveau style. C'est-à-dire que si un agent d'utilisateur qui suit à la fois la présente spécification et la spécification originale de Netscape reçoit un en-tête de réponse Set-Cookie2, et que le NOM et les attributs Domaine et Chemin correspondent (selon le paragraphe 3.3.2 Gestion du mouchard) un mouchard de style Netscape, celui-ci DOIT être éliminé, et l'agent d'utilisateur DOIT ne retenir que le mouchard qui adhère à la présente spécification.

Les plus anciens agents d'utilisateur qui ne comprennent pas la présente spécification, mais comprennent la spécification originale de Netscape, ne vont pas reconnaître l'en-tête de réponse Set-Cookie2 et vont recevoir et envoyer des mouchards conformément à l'ancienne spécification.

Un agent d'utilisateur qui prend en charge à la fois les mouchards de la présente spécification et ceux de style Netscape DEVRAIT envoyer un en-tête de demande Cookie qui suit l'ancienne spécification Netscape si il reçoit le mouchard dans un en-tête de réponse Set-Cookie et non dans un en-tête de réponse Set-Cookie2. Cependant, il DEVRAIT envoyer l'en-tête de demande suivant aussi :

```
Cookie2: $Version="1"
```

L'en-tête Cookie2 avise le serveur que l'agent d'utilisateur comprend les mouchards de nouveau style. Si le serveur comprend les mouchards de nouveau style aussi, il DEVRAIT continuer la session à états pleins en envoyant un en-tête de réponse Set-Cookie2, plutôt qu'un Set-Cookie. Un serveur qui ne comprend pas les mouchards de nouveau style va simplement ignorer l'en-tête de demande Cookie2.

9.2 Mise en antémémoire et HTTP/1.0

Certaines antémémoires, comme celles qui se conforment à HTTP/1.0, vont inévitablement mettre en antémémoire les en-têtes Set-Cookie2 et Set-Cookie, parce qu'il n'existe pas de mécanisme pour supprimer la mise en antémémoire des en-têtes avant HTTP/1.1. Cette mise en antémémoire peut conduire à des problèmes de sécurité. Les documents transmis par un serveur d'origine avec les en-têtes Set-Cookie2 et Set-Cookie ne peuvent usuellement être ni empêchés d'être mis en antémémoire ni être "pré-expirés". Tant que les antémémoires obéissent aux instructions de ne pas mettre les documents en antémémoire (suivant Expires: <une date passé> ou Pragma: no-cache (HTTP/1.0), ou Cache-control: no-cache (HTTP/1.1)) les documents qu'on ne peut pas mettre en antémémoire ne présentent pas de problème. Cependant, les documents pré-expirés peuvent être mémorisés dans des antémémoires. Ils exigent une validation (un GET conditionnel) sur chaque nouvelle demande, mais certains opérateurs d'antémémoires relâchent les règles régissant leurs antémémoires, et servent parfois des documents arrivés à expiration sans d'abord les valider. Cette combinaison de facteurs peut conduire à ce que des mouchards destinés à un usager soient ultérieurement envoyés à un autre usager. Les en-têtes Set-Cookie2 et Set-Cookie sont mémorisés dans l'antémémoire et, bien que le document soit périmé (arrivé à expiration) l'antémémoire retourne le document en réponse à des demandes ultérieures, y compris les en-têtes mis en antémémoire.

10. Remerciements

Le présent document représente réellement l'effort collectif du groupe de travail HTTP de l'IETF et, en particulier, des personnes suivantes, en plus des auteurs : Roy Fielding, Yaron Goland, Marc Hedlund, Ted Hardie, Koen Holtman, Shel Kaphan, Rohit Khare, Foteos Macrides, David W. Morris.

11. Adresse des auteurs

David M. Kristol
Bell Laboratories, Lucent Technologies
600 Mountain Ave. Room 2A-333
Murray Hill, NJ 07974 USA
téléphone : (908) 582-2250
fax : (908) 582-1239
mél : dmk@bell-labs.com

Lou Montulli
Epinions.com, Inc.
2037 Landings Dr.
Mountain View, CA 94301
USA
mél : lou@montulli.org

12. Références

[DMK95] Kristol, D.M., "Proposed HTTP State-Info Mechanism", disponible à <http://portal.research.bell-labs.com/~dmk/state-info.html>, septembre, 1995.

[Netscape] "Persistent Client State -- HTTP Cookies", disponible à <http://www.netscape.com/newsref/std/cookie_spec.html>, sans date.

[RFC2109] D. Kristol, L. Montulli, "Mécanisme de gestion d'état HTTP", février 1997. (*Obsolète, voir* RFC2965) *(P.S.)*

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2279] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", janvier 1998. (*Obsolète, voir* RFC3629) *(D.S.)*

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

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

13. Déclaration de droits de reproduction

Remerciement

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