Webh4ck partie IV

© Stephane Rodriguez, 12 Novembre 2000

Ce n'est pas le tout de se donner les moyens de traverser des pages web, des formulaires, bref des sites entiers de manière totalement automatique, il faut ici en faire quelque chose. Et les applications basées sur ce principe supposent l'utilisation ou non d'un navigateur web.

Cela veut dire qu'à partir du moment où la traversée d'un site, au sens général du terme, n'est là que pour donner une page à un navigateur web, il est évidemment hors de question que l'outil de traversée soit une moulinette dédiée installée côté client et incapable d'intéragir avec le navigateur web.

Il faut qu'il y ait symbiose.

Web4ck peut être un logiciel applicatif dédié côté client, un proxy local côté client, un code serveur derrière un serveur web classique (approche script, approche ISAPI, approche DLL), un serveur dédié. Ou une combinaison de tout cela….

Une façon simple de savoir comment doit être implémenté Webh4ck en pratique est d'abord de se demander à quoi il va servir. Ceci imposera probablement certaines contraintes.

Première utilisation : métarecherche
Seconde utilisation : log automatique sur un compte webmail
Troisième utilisation : capture de news pour lecture offline
Quatrième utilisation : accès direct à une page (sans login)
Cinquième utilisation : remplissage automatique d'un formulaire
…et de nombreuses autres qui ne demandent qu'à être imaginées…..Le web est un vrai réservoir d'applications….

1.# Quelques utilisations de Webh4ck

1.1# Métarecherche

La métarecherche consiste à exploiter plusieurs moteurs de recherche en simultané. Ce n'est d'ailleurs pas très utile, puisque chaque moteur de recherche est susceptible à lui seul de remonter des centaines de milliers de pages compatibles avec ce qui est recherché, au même titre que tous les autres moteurs de recherche, sans parler du problème évident de la redondance.

Typiquement, mon mot-clef est post-it, je souhaite savoir qui parle de post-it sur le web. Je vais sur www.yahoo.fr, je tape le mot-clef et lance la recherche. J'ai validé un formulaire. Comme une page web ne suffit souvent pas à contenir tous les URLs pertinentes, les moteurs de recherche mettent en œuvre une approche incrémentale dans laquelle chaque page web retournée contient un lien vers la page web suivante. En fait de lien vers la page web suivante, il s'agit de nouveau de la requête initiale issue lors de l'envoi du formulaire, mais avec un paramètre indice permettant d'avancer dans la liste des résultats. Il faut savoir que les moteurs de recherche ne font pas une recherche en temps réel, ils ne font qu'exploiter d'immenses bases de données construites incrémentalement par des robots logiciels qui fonctionnent nuit et jour.

La recherche sur un seul moteur est simple, et déjà plusieurs applications peuvent en être retirées.

La première, pourquoi passer par la homepage de yahoo pour lancer la recherche. N'est-ce pas plus simple dans le contexte de mon espace de travail de taper un mot-clef dans une zone de saisie, et de choisir un moteur de recherche ? On sait par ailleurs qu'il est tout à fait pertinent de choisir soi-même un moteur de recherche puisque certains sont réputés pour jouer aussi le rôle d'annuaire, ou même indexer les données du web de façon très

particulière, ce qui contribue à faire d'eux des moteurs de recherche privilégiés. Ce n'est pas pour rien que Yahoo a un tel succès !

La seconde application consiste à lancer une recherche mais bénéficier d'une présentation différente des résultats. Une sorte d'intermédiaire reformulerait les résultats, sur le plan de la présentation visuelle (HTML), ou sur le plan du classement, ou autre, de façon à optimiser l'exploitation.

La troisième application consiste à exploiter de façon complète les paramètres techniques mis en jeu lors d'une recherche. Jouer sur les paramètres permet d'optimiser une recherche. On peut imaginer un intermédiaire capable de fixer certains paramètres, équivalent d'une recherche avancée dans Yahoo, voire de faire ce que l'interface Yahoo ne permet pas de faire. Impressionnant et domaine très ouvert, même si tout le monde risque de ne pas apprécier cette perspective !

La métarecherche consiste à exploiter plusieurs moteurs à la fois, à partir du même mot-clef. Prise de tête quasi garantie mais c'est le principe de pouvoir le faire qui est intéressant. Au delà du principe, il est assez intéressant de se poser la question du recoupement des informations obtenues séparément, et de l'élimination des redondances. Tout un programme !

1.2# Log automatique sur un compte webmail

Quoi de plus rébarbatif que d'être obligé de rentrer un login et un mot de passe pour accéder à un compte email sur le web, comme sur hotmail ? Bien sûr, c'est logique pour éviter que n'importe qui aille sur mon compte, se fasse passer pour moi, lise mes emails, en envoie, etc. Mais lorsque rentrer son login et mot de passe est synonyme d'actions répétées, par exemple dix fois par jour, il est souhaitable de disposer d'un accès direct au compte.

Et quelle immanquable opportunité de constituer un outil d'agrégation de comptes emails multiples. Il suffit d'imaginer que par exemple on puisse sur une page web obtenir la totalité des emails reçus sur chacun de mes comptes : ceux sur hotmail, sur iname, sur … L'application peut être purement consultative, et on est prêt à se logguer réellement sur un compte email pour répondre. Mais l'application peut aussi être beaucoup plus intelligente et remonter par exemple uniquement les nouveaux emails reçus, voire pourquoi pas donner la possibilité d'intéragir et d'y répondre, comme si on était dans le compte email en question. On le voit, il y a une place ici pour une application qui casserait l'isolation des comptes email sur le web et qui permettrait de gagner beaucoup de temps.
Couplé à un système d'alerte, il n'est plus nécessaire de chercher à savoir soi-même si de nouveaux emails sont arrivés, et ce quel que soit le compte sur l'ensemble des comptes ouverts.
L'information vient à vous. De cette masse gigantesque d'informations que représente Internet, des millards et des milliards de mots, vous êtes "abonné" uniquement à ce qui compte vraiment.

1.3# Capture de news pour lecture offline

Internet fourmille de portails de news. A tel point que quelqu'un qui chercherait à savoir tout ce qui se dit dans le monde entier n'aurait pas assez de 24 heures pour accomplir sa tâche. Seul problème et de taille, 24 heures, c'est le maximum de temps que l''on octroye à quelqu'un avant qu'un portail rafraichisse ses données ! Tâche insurmontable donc.

C'est là qu'intervient un outil de capture de news. Il suffit d'imaginer que chaque portail majeur (infos générales, météo, trafic, informatique, news américaines, …) soit indexé une fois pour toutes, c'est-à-dire que l'on y repère son bloc de news au sein de la page, puis qu'une moulinette fasse quotidiennement un travail de capture de chacun de ces blocs et constitue un résumé agrégé dans une page web unique.
Il suffit pour cela de sa faire à l'idée qu'avec un tel portail, il suffit de quelques minutes quotidiennes pour savoir ce qui se passe dans le monde entier. Une fois que l'on sait ce qui se fit, l'instinct de "surfer", de savoir ce qui se passe, est beaucoup moins fort, et l'employé d'une entreprise est de fait plus enclin à se consacrer à sa tâche, celle

pour laquelle il est payée ! Il suffit de mettre à disposition sur l'intranet de l'entreprise la page web obtenue par la moulinette de capture pour partager l'information avec un grand I.

1.4# Accès direct à une page (sans login)

Cette application rejoint en partie celle permettant d'accéder aux comptes emails.

Imaginons que j'ai un compte sur un site web. Imaginons que pour administrer mon compte je suis obligé de rentrer mon login et mon mot de passe, puis traverser des pages pour arriver sur une page qui m'intéresse. Je sais d'avance que c'est cette page qui m'intéresse, et je sais d'avance qu'après m'être loggué, je vais cliquer sur ce lien puis sur ce lien, etc. pour arriver à cette page.
Tout ceci est très rébarbatif, alors qu'il est louable de pouvoir faire la même chose d'un simple clic !!
La différence essentielle avec les comptes emails web, c'est qu'au lieu de "capturer" les emails et de les présenter dans une nouvelle page web générée à la volée, ici l'idée consiste à fournir un mécanisme de traversée automatique pour aller sur la page qui nous intéresse. Nul besoin ici une fois la page obtenue, de capturer certaines choses et de renvoyer une page web ainsi constituée.
C'est donc une toute autre application et c'est tout aussi intéressant, car synonyme de gain de temps, et de fin de tâches on ne peut plus répétitives !

Le bémol qu'on peut mettre là dessus est que les administrateurs de sites ont tendance à réorganiser la structure de ceux-ci tous les deux ou trois mois en moyenne. Et qui restructuration dit changement des paramètres, changement des composants de traversée, etc. En conséquence, un outil de traversée, pour automatique qu'il soit, doit être fréquemment mis à jour.
Mais même pour cela, des robots logiciels peuvent régulièrement simuler des connections et vérifier que des paramètres attendus sont toujours disponibles et/ou que la page attendue est toujours accessible. Toute erreur indique soit une panne du serveur, soit précisément un changement de structure. Avec un système d'alerte, à la fois l'utilisateur final, qui lui ne se doute pas de toute cette cuisine, et l'administrateur de Webh4ck ont en main des outils essentiellement automatisés, et ne perdent pas de temps en vérification manuelle.

Il faut noter que les administrateurs de sites rivalisent d'astuces afin de rendre difficile la dite traversée automatique d'un site, et ce grâce aux artifices connus : les cookies, les variables de session et autres entêtes HTTP invisibles. Webh4ck prend tout cela en compte.

1.5# Remplissage automatique d'un formulaire

Quoi de plus rébarbatif que de remplir *n* fois un formulaire ? On a vu l'exemple d'un formulaire simplifié, celui mis en œuvre lors d'une recherche sur le web, mais de nombreux formu laires parcourent la navigation sur le web aujourd'hui : formulaire de login, d'enregistrement nominatif, d'achat, d'envoi de suggestion, ….

Pourquoi ne pas enregistrer un formulaire très complet une fois pour toutes et "jeter" ces données par drag&drop à chaque fois qu'un nouveau formulaire se présente ?

Ceci est faisable. Et que dire de celle qui consiste à s'enregistrer sur un nombre conséquent de sites ?
S'enregistrer, kesako ? imaginons que l'on veuille s'enregistrer automatiquement à toutes les newsletters (courrier électronique quotidien ou hebdomadaire). Imaginons que l'on veuille déposer son CV sur plein de sites, automatiquement. Bien pratique !
Déposer automativement son CV sur une ribembelle de sites d'emploi, c'est fournir à l'utilisateur lamb da un moyen de gagner un temps incommensurable d'une part, et de lui éviter un certain nombre de tracas techniques dont on se passe très bien d'autre part. En ce sens, cette application est "par-dessus" les sites d'emploi permettant individuellement de déposer un CV. On ne casse rien à la logique, on ajoute une couche supplémentaire. Cette

couche permet tout aussi facilement de vérifier que le CV a été enregistré, de supprimer son CV dès que le candidat a trouvé chaussure à son pied, etc.

2.# Point commun entre les applications issues de Webh4ck

On constate une chose, c'est que les applications de Webh4ck visent à simplifier et optimiser l'accès au web d'une manière ou d'une autre. Mais il s'agit encore et toujours de web.
Il est donc assez naturel qu'une application issue de Webh4ck fournisse une interface d'administration elle-même pur web. En d'autres termes, l'accès à un ensemble de moteur de recherche doit aussi le plus naturellement possible se faire par le biais d'une page web, et non d'une application client dédiée.

Cela dit, la tentation est grande de laisser faire des moulinettes, sortes de boites noires, tourner en toile de fond quelques minutes par jour, ou toute la journée, et recueillir une fois par jour le fruit de leur recherche à travers une page web générée à la volée, une sorte d'accusé de réception électronique.

En termes techniques, cela revient à se poser la question de savoir quel composant assure la transport des requêtes web HTTP et la réception des pages web ? Est-ce le navigateur web de l'utilisateur, est-ce une librairie HTTP/HTML disponible sur un serveur d'application, est-ce un composant bas niveau utilisant les sockets, bref une librairie HTTP/HTML côté client ?

Les contraintes de navigation reviennent à être capable de simuler :
- des requêtes GET, issues d'un clic sur un lien hypertexte d'une page web
- des requêtes POST, issues de la validation des données d'un formulaire
- l'envoi de données invisibles tels les cookies ou paramètres de session

Par définition, un navigateur web est capable de simuler les trois puisque c'est ce que l'on fait quand on surfe sur le web ! Mais il faut bien comprendre SIMULER.

3.# Base de données

Webh4ck est par exemple censé être capable d'automatiser l'accès à des pages en traversant un formu laire de login. Et ce jour après jour. Cela veut dire qu'il faut avoir préalablement indexé le formulaire de façon unique, et la page cible. Données dont se sert Webh4ck. Ces données doivent être stockées sur la machine où s'exécute Webh4ck.
Mais rien n'impose que ces données soit dans une base de données type Access , SQL Server, Oracle, … En fait, les données peuvent être agrégées dans un fichier. Mais l'opportunité est forte d'utiliser un fichier de description XML. Car utiliser XML aujourd'hui, c'est s'assurer un moyen simple de transvaser les données depuis et vers une base de données, plus tard, si le besoin s'en fait sentir.
Si les données étaient stockées dans un fichier plat, un fichier propriétaire, alors il faudrait développer une procédure d'importation et d'exportation. Inutile.
Mieux, le fichier de description XML peut être obtenu par extraction de certaines données d'un autre fichier XML.
Microsoft fournit gratuitement un analyseur XML : **MS XML 3.0 SDK**. Cet analyseur permet à la fois de créer des fichiers structurés XML à partir de données en mémoire, et de décoder un fichier XML structuré.

La partie I de Webh4ck aborde la structure de données retenue pour modéliser la navigation automatique sur le web. Elle est basée sur le fondement du web, à savoir un ensemble de paramètres GET, POST et des entêtes invisibles comme les cookies.

Proposition de structure XML :

```
<url name="page1" source="http://127.0.0.1/" port="80" method="GET">
        <header name="…" value="…"/>
```

```
        …
        <get name="…" value="…"/>
        …
        <post name="…" value="…"/>
        …
        <declare name="set-cookie"/>
</url>
<url name="page2" source="http://127.0.0.1/login.asp"  port="80" method="GET" output="text.html">
        <header name="cookie" value="page1:set-cookie[0]"/>
        …
        <get name="…" value="…"/>
        …
        <post name="…" value="…"/>
        …
</url>
```

Explications :

- le fichier ci-dessus détaille les paramètres d'accès à deux pages web distinctes.
- L'accès à la page 1 est à préalable à la page 2. Elle permet d'obtenir un identifiant par le biais d'un cookie
- Seul l'obtention de ce cookie permet un accès au contenu de la page 2, ce qui justifie la précédence d'un des attributs en page 2
- En page 1, on déclare une URL accédée par une méthode GET, équivalent à un clic sur un lien hypertexte
- En page 1, on déclare une liste d'entêtes, ensuite transcrites de la façon suivante nom_entête: valeur_entête. Il y a un nombre arbitraire d'entêtes, éventuellement aucune.
- En page 1, on déclare une liste de paramètres GET. Chaque paramètre GET sera concaténé à l'URL de base comme dans http://127.0.0.1?login=steph&pwd=rod. Il y a un nombre arbitraire de paramètres GET, éventuellement aucun.
- En page 1, on déclare une liste de paramètres POST. Chaque paramètre est concaténé dans une chaine, au même titre que les paramètres GET, mais sont finalement déposé dans le corps de la requête, après les entêtes. Il y a un nombre arbitraire de paramètres POST, éventuelle ment aucun.
- En page 1, on déclare vouloir récupérer la valeur d'une entête portant le nom "set-cookie"
- En page 2, on déclare une nouvelle URL et des entêtes, des paramètres GET et POST
- En page 2, à la différence de la page 1, une des entêtes transmises comporte une valeur obtenue dynamiquement. page1:set-cookie se réfère à la syntaxe nom_url:nom_entête. De plus, [0] indique la première occurrence de cette entête, car la page 1 peut recevoir un nombre arbitraires de cookies. [1] permet par ce principe d'accéder au second cookie.
- En page 2, de même qu'il est possible d'accéder à une entête reçue, il est possible de récupérer la valeur d'un paramètre GET ou POST ou d'une entête envoyée.
- En page2, l'attribut OUTPUT permet de générer un fichier HTML sur disque.

La DTD initiale à ce fichier XML est :

DTD de Webh4ck :

```
<?xml version="1.0" encoding="UTF-8" ?>
<!- Fichier de description Webh4ck ->
<!DOCTYPE webh4ck [
        <!ELEMENT url (header | get | post | declare)* >
        <!ATTLIST url
                name     CDATA #IMPLIED
                source   CDATA #REQUIRED
                port     CDATA #IMPLIED
                method   (GET | POST) "GET"
                output    CDATA #IMPLIED >
        <!ELEMENT header EMPTY>
        <!ATTLIST header
```

```
                name    CDATA #REQUIRED
                value   CDATA #REQUIRED>
        <!ELEMENT get EMPTY>
        <!ATTLIST get
                name    CDATA #REQUIRED
                value   CDATA #REQUIRED>
        <!ELEMENT post EMPTY>
        <!ATTLIST post
                name    CDATA #REQUIRED
                value   CDATA #REQUIRED>
        <!ELEMENT declare EMPTY>
        <!ATTLIST declare
                name    CDATA #REQUIRED>
]>
```

(La DTD ne définit pas à elle seule toute la logique applicative, notamment en ce qui concerne le passage de valeurs d'entêtes d'une page à l'autre).

La logique de passage de paramètre entre deux pages à ses limites. En effet, il suffit que l'entête à envoyer soit constituée non pas de la valeur exacte d'une autre entête mais d'une combinaison plus compliquée, mettant en jeu d'autres paramètres et d'autres caractères, pour que la logique ne fonctionne pas.
Il faut envisager alors une syntaxe plus générale que **page1:set-cookie[0]**. Traiter le cas général revient à mettre du code fonctionnel dans la syntaxe.

Un cas qui revient souvent est la concaténation de plusieurs champs, certains statiques, certains dynamiques.
On peut imaginer une syntaxe de la forme **CONCAT('login=arstlg&sessionid=',page1:set-cookie[0],'&timeout=15')**

4.# Gradation des difficultés

Même si Webh4ck sait fondamentalement accéder à n'importe quelle page web, le cas général peut être obtenu par combinaison de cas élémentaires. Ces cas élémentaires permettent de concevoir les composants Webh4ck en fonction des besoins réels.
Par exemple, l'accès le plus simple à une page est concrétisé par une URL sans paramètre GET. L'exemple qui vient tout de suite après est le cas d'une page comportant un paramètre GET. Puis plusieurs paramètres.
Idem pour les paramètres POST.
Idem pour les entêtes invisibles.

Il suffira ensuite de combiner tous ces cas de figure pour obtenir le cas général.

Cela dit, accéder à une page web peut nécessiter l'accès préalable à une autre page web, d'où un enchainement. Mais le principe de base est inchangé.

4.1# Page web simple

4.2# Page web avec un paramètre GET

4.3# Page web avec plusieurs paramètres GET

4.4# Page web avec un paramètre POST

4.5# Page web avec plusieurs paramètres POST

4.6# Page avec un entête invisible

4.7# Page avec plusieurs entêtes invisibles

4.8# Combinaison

4.9# Chaînage de pages

La difficulté du chaînage de pages provient de la récupération des données obtenues à la ou les pages précédentes. La question qui se pose est de savoir si les données à récupérer sont dont la partie entête retournée par le serveur, ou dans le corps, voire dans des paramètres GET/POST d'une page précédente (ou future).

Cas de la partie entête :
Une syntaxe particulière au niveau XML permet de déclarer les entêtes "sources". Et une syntaxe doit pouvoir se référer à la valeur prise par cette entête. L'exemple précédent **page1:set-cookie** fait référence à une entête retournée lorsque l'on accède à la page 1. Cette entête est déclarée dans l'élément page1, de façon à pouvoir contrôler la cohérence. Cette déclaration n'est pas obligatoire. De plus, comme le serveur est susceptible de retourner 0, 1 ou n entêtes portant ce nom, il faut résoudre explicitement le conflit potentiel. Une notation de type accès-tableau suffit.
Ce même principe s'applique pour récupérer la valeur d'un paramètre GET ou POST de n'importe quelle page web définie dans le fichier XML, et pas nécessairement une page déjà vue. Tout est question de séquencement.

Cas du corps :
Le corps est potentiellement et essentiellement un code HTML , c'est-à-dire une structure complexe. Récupérer une donnée incluse dans un code HTML suppose la traversée de la structure arborescente de tags. Ceci se fait par l'intermédiaire d'un API de programmation D.O.M. (document object model).

<SYNTAXE A DEFINIR>
WalkAll, un produit annexe, définit un chemin d'accès à un tag. Cette notation peut être utilisée tel quel.
La notation issue de la norme W3C Xpath peut également être exploitée.