Université catholique de Louvain

Travail de fin d'études

Antoine Marchal

Promoteurs :
Olivier Bonaventure
Pierre Reinbold

Lecteurs :

Mémoire réalisé en vue de l'obtention du grade de
Master [120] en Sciences Informatiques (option *Networking & Security*)

Version temporaire, compilée le 26 avril 2014

Table des matières

Chapitre 1

Introduction

1.1 Motivation et problématique

Notre navigation sur Internet est de plus en plus scrutée par diverses organisations, que ce soient des entreprises commerciales ou des agences gouvernementales. D'ailleurs, des révélations récentes [1] ont permis de mettre en lumière un système élaboré destiné à surveiller les utilisateurs d'Internet.

Ce travail se focalise et analyse la surveillance opérée par les entreprises commerciales. En effet, elles essaient de nous suivre à la trace afin de déterminer nos habitudes et nos préférences. Avec l'aide de ces précieuses informations, elles peuvent alors mieux cibler nos besoins et nous proposer des biens et des services qui sont censés nous intéresser davantage, voire anticiper nos intentions [2].

Cette pratique est de plus en plus répandue sur les sites de vente en ligne, grâce à l'historique de nos achats.

D'autres formes de surveillance plus vicieuses existent également. Imaginons que vous souhaitiez voyager vers un pays exotique et que vous consultez le prix des billets d'avion vers cette destination sur le site d'une compagnie aérienne. Lors d'une consultation ultérieure, le prix a augmenté et vous vous empressez alors d'acheter le billet craignant que son prix ne continue d'augmenter. Malheureusement pour vous, celui-ci a en fait été gonflé pour vous pousser à l'achat.

Les plus grosses régies publicitaires sont installées sur un grand nombre de sites et lors de chaque visite, elles enregistrent les traces que vous laissez. En regroupant toutes les informations disséminées sur ces différents sites, elles sont alors en mesure d'établir votre profil. Les détails de celui-ci peuvent se revendre à prix d'or auprès d'autres compagnies. Il est évidemment plus facile de cibler un segment consitué de personnes dont on connaît le profil socio-démographique, le sexe, l'âge, les intérêts,...

Avez-vous déjà remarqué que le contenu des publicités s'adaptait en fonction de vos requêtes effectuées sur les sites de recherche ou des sites que vous visitez ?

Comme vous pouvez le remarquer avec ces différents exemples de la vie quotidienne, la surveillance des utilisateurs d'Internet peut amener à plusieurs inconvénients qui nous touchent directement. Voici donc pourquoi il est intéressant de regarder à cette surveillance et d'essayer de déterminer via quels moyens celle-ci est opérée. Afin de préserver notre vie privée sur Internet, il est nécessaire de connaître comment nous sommes identifiés. Il faut également s'assurer que les outils dont nous disposons afin de nous protéger soient réellement efficaces et adaptés à nos besoins.

1.2 Pourquoi vouloir l'anonymat sur Internet ?

1.3 Méthodologie utilisée

Afin de répondre à cette problématique, il faut tout d'abord comprendre comment Internet fonctionne et en particulier, certains de ses mécanismes tels que les cookies.

Ensuite, il faut identifier la manière dont ces mécanismes peuvent être utilisés ou détournés afin de permettre le traçage des utilisateurs.

Grâce à ces connaissances, nous serons en mesure de détecter quels sont les moyens effectivement utilisés par les sites qui pratiquent cette surveillance et d'estimer son étendue sur un panel constitué des principaux sites web (classement Alexa [3]).

Pour terminer, nous pourrons déterminer si les outils censés protéger notre vie privée sont réellement efficaces.

Chapitre 2

Protocoles et mécanismes d'Internet

2.1 Présentation du protocole HTTP

HTTP (HyperText Transport Protocol) [4] est un protocole qui fournit les fondations du World Wide Web, il repose sur le modèle client-serveur dans lequel le client envoie une requête et le serveur retourne une réponse.

Les requêtes HTTP se composent de trois parties :

1. Une ligne de requête (qui contient notamment une URI)
2. Un en-tête (qui contient des paramètres facultatifs pour la requête)
3. Le corps de la requête (optionnel)

Lorsqu'un utilisateur clique sur un lien hypertexte via son navigateur, celui-ci se connecte à un serveur identifié par l'URI "Uniform Resource Identifier" présente dans ce lien. Le navigateur envoie alors une requête à laquelle le serveur répond puis le navigateur se déconnecte du serveur. L'on considère que la requête est sans état car à chaque fois que le navigateur crée une connexion pour une requête, le serveur la traite comme si c'était la première ; les requêtes sont donc indépendantes.

Les réponses HTTP se composent également de trois parties :

1. Une ligne de statut (qui indique si la requête est réussie ou pas)
2. Un en-tête (qui contient des informations additionnelles sur la réponse)
3. Le corps de la réponse

Il fallait trouver une solution afin d'avoir la possibilité de garder une certaine quantité d'informations entre des requêtes successives. Une première solution était de forcer l'authentification du client (comme pour FTP), cependant ce n'est pas toujours nécessaire ou applicable sur tous les sites web. Une seconde solution était d'utiliser les différents types d'en-têtes *Accept-**, mais cela fournit des possibilités assez limitées. Une autre solution, qui est la plus largement adoptée, est l'utilisation d'un cookie HTTP.

Des exemples de requêtes et réponses HTTP sont disponibles dans la section Cookies.

2.2 Cookies

La première description sur les cookies a été publiée sur le site web de Netscape Communications mais cette description était informelle. Le processus de standardisation des cookies a commencé en avril 1995 sur la liste de diffusion [www-talk], ensuite, l'IETF (Internet Engineering Task Force) a entrepris d'écrire un standard pour les cookies [5]. C'est ainsi que la première RFC (RFC 2109) sur les cookies est parue en février 1997 [6], rendue obsolète en octobre 2000 par la RFC 2965 [7]. Elle a également été rendue obsolète par la dernière RFC qui est toujours d'application en ce jour, la RFC 6265 (avril 2011) [8].

Un cookie est un petit fichier stocké en clair sur le disque dur de l'utilisateur par le navigateur. Il fait le lien entre la session de l'utilisateur et les données enregistrées par le site web (dans une base de données par exemple). Les cookies sont transmis dans les en-têtes des requêtes et des réponses HTTP [8].

Dans sa réponse à une requête, un serveur peut envoyer des informations arbitraires (le cookie) dans un en-tête *Set-Cookie*. Cette information peut contenir n'importe quoi et c'est elle qui permet au serveur de continuer là où il en était. Il peut s'agir d'un identifiant relatif à l'utilisateur, une clé dans une base de données,...

Habituellement, le client est coopératif et renvoie l'information du cookie qui est stocké par le navigateur. Dans chaque requête ultérieure qu'il fait vers le serveur, le client indique cette information dans un en-tête *Cookie*. Le serveur peut choisir de renvoyer un nouveau cookie dans ses réponses, ce qui remplacera automatiquement l'ancien cookie.

Il y a un contrat implicite entre le client et le serveur : ce dernier repose sur le client afin de sauvegarder son état et il espère qu'il lui sera retourné lors de la prochaine requête. Un cookie est donc une donnée que le serveur et le client se renvoient l'un et l'autre. La quantité d'informations est généralement petite et son contenu est à la discrétion du serveur. En effet, la plupart du temps, analyser le contenu du cookie ne révèle ni à quoi il est destiné ni la valeur qu'il représente.

Voici une requête HTTP vers le site web *distrowatch.com* :

```
1 GET / HTTP/1.1
2 Host: distrowatch.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:28.0) Gecko
    /20100101 Firefox/28.0
4 Accept: text/html,application/xhtml+xml,application/xml;q
    =0.9,*/*;q=0.8
5 Accept-Language: fr-be,en-us;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Connection: keep-alive
```

Et la réponse HTTP associée :

```
1 HTTP/1.1 200 OK
2 Date: Sat, 26 Apr 2014 12:32:09 GMT
3 Server: Apache/2.2.22 (Debian)
4 X-Powered-By: PHP/5.4.4-14+deb7u8
5 Expires: Mon, 26 Jul 1997 05:00:00 GMT
6 Last-Modified: Sat, 26 Apr 2014 12:32:09 GMT
```

```
7  Cache-Control: no-store, no-cache, must-revalidate, post-check
       =0, pre-check=0
8  Pragma: no-cache
9  Set-Cookie: NewLastSeen=8407; expires=Tue, 06-May-2014 11:32:09
       GMT
10 Vary: Accept-Encoding
11 Content-Encoding: gzip
12 Content-Length: 33808
13 Keep-Alive: timeout=15, max=100
14 Connection: Keep-Alive
15 Content-Type: text/html
```

Si l'on retourne sur le site, la requête HTTP contient maintenant un header *Cookie* visible à la ligne 7 :

```
1  GET / HTTP/1.1
2  Host: distrowatch.com
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:28.0) Gecko
       /20100101 Firefox/28.0
4  Accept: text/html,application/xhtml+xml,application/xml;q
       =0.9,*/*;q=0.8
5  Accept-Language: fr-be,en-us;q=0.7,en;q=0.3
6  Accept-Encoding: gzip, deflate
7  Cookie: NewLastSeen=8407; __utma
       =63506106.1883630997.1398515530.1398515530.1398515530.1;
       __utmb=63506106.1.10.1398515530; __utmc=63506106; __utmz
       =63506106.1398515530.1.1.utmcsr=(direct)|utmccn=(direct)|
       utmcmd=(none)
8  Connection: keep-alive
```

Quant à la réponse HTTP, elle reste identique.

2.3 Cache

Dans le domaine informatique, un cache permet de garder une copie locale d'un élément afin de répondre rapidement à une requête. Au lieu de récupérer l'élément, le cache renvoie sa copie, ce qui permet de réduire sensiblement le temps de réponse.
Sur Internet, plusieurs mécanismes de cache ont été implémentés [9]. Ils peuvent par exemple être installés à différents niveaux :

- sur le réseau qui délivre le contenu d'un site web (ex. : un CDN - "Content Delivery Network")
- sur l'application qui gère et affiche le contenu d'un site web (ex. : un CMS - "Content Management System")
- sur le serveur qui héberge le site web (ex. : Apache)
- via le navigateur Web du client qui stocke les fichiers sur le disque dur

Lorsque que l'on navigue sur Internet, le cache du navigateur enregistre certaines ressources (images, feuilles de style, fichiers JavaScript, etc.) afin de ne pas devoir les recharger depuis le serveur lors des visites ultérieures. Par ailleurs, il est possible pour les

administrateurs d'un site web de paramétrer et de limiter la mise en cache de certaines pages grâce à l'en-tête *Cache-Control*.

2.4 Comment le Web est-il censé fonctionner ?

Lors du chargement d'une page web, le navigateur effectue de multiples connexions afin de récupérer l'ensemble des ressources de la page. Aux débuts du World Wide Web, toutes les ressources d'une page appartenaient généralement à une même personne/groupe. A l'heure actuelle, la situation a fort changé : il est fréquent de voir des ressources chargées depuis des domaines tiers (régies publicitaires, CDN,...). Le navigateur effectue donc des connexions vers des serveurs différents (cela permet notamment de passer outre la limitation du nombre de connexions HTTP vers un même domaine [10]) mais en contrepartie, cela permet aux serveurs en question d'enregistrer des données sur le client suite aux requêtes qu'il effectue.

Lors du développement de Netscape Navigator 2, une décision importante a été prise concernant le principe de même origine ("same-origin" principle) [11]. Celui-ci interdit des sites web de domaines différents d'interagir entre eux au niveau du cache du navigateur, sauf dans des cas précis. Ce principe est implémenté dans les différents navigateurs mais il n'est appliqué de la même manière. L'échec de l'adaptation de ce principe semble être la source la plus importante de fuites de données.

Chapitre 3

Moyens d'identification

3.1 Cookies

3.2 Cache

Le cache du navigateur permet d'enregistrer localement une copie des fichiers afin de ne pas devoir les recharger lors d'une visite ultérieure. Ce mécanisme est très utile car il permet d'économiser de la bande passante et du temps, cependant il donne la possibilité à des sites web de déterminer si leurs visiteurs ont visité un autre site auparavant.

L'exploitation du cache à des fins de surveillance est détaillé dans [12]. Le principe est assez simple : il exploite le fait qu'un fichier présent en cache sera chargé beaucoup plus rapidement qu'un fichier qui ne l'est pas. Donc en mesurant le temps d'accès au fichier, il est possible de déterminer si une personne a déjà visité le site web (ou plus précisement, la page) qui utilise ce fichier. En effet, rien n'empêche un site web de charger un fichier hébergé par un autre site.

Imaginons que l'administrateur d'un site *(alpha.com)* veuille savoir si ses visiteurs se sont également rendus sur un autre site *(beta.com)*. La première chose qu'il doit faire est de se rendre sur le site qu'il veut cibler et choisir un fichier statique pouvant être mis en cache et qui est chargé par tout visiteur (un logo par exemple). Ensuite, le but est de mesurer le temps d'accès du fichier cible.

D'après [12], le plus fiable et facile est d'utiliser un applet Java ou un script écrit en JavaScript qui vont mesurer le temps de chargement du fichier à partir de son URL. Même si l'utilisateur a désactivé l'utilisation de Java ou JavaScript, il est possible d'obtenir une mesure suffisament précise en chargeant les fichiers suivants dans l'ordre :

1. un fichier du site *alpha.com*
2. le fichier cible du site *beta.com*
3. un autre fichier du site *alpha.com*

En soustrayant les moments auxquels le serveur reçoit les requêtes des fichiers 1 et 3, l'administrateur du site *alpha.com* est en mesure d'avoir une approximation du temps qu'il a fallu pour charger le fichier 2 (celui de *beta.com*).

Il faut néanmoins respecter certains critères pour que cela fonctionne : forcer le chargement des fichiers de manière séquentielle et de manière invisible en n'altérant pas l'apparence de la page afin que le client ne remarque rien.

Il est possible d'améliorer la probabilité de distinguer correctement les succès des défauts de cache en effectuant différentes mesures, ce qui permet alors de rafiner les seuils de discrimination : refaire plusieurs fois la mesure d'un même fichier (à partir de la seconde tentative, le fichier sera présent dans le cache) pour les succès de cache et utiliser l'URL de fichiers n'existant pas pour les défauts de cache.

Afin d'améliorer la précision, il est également intéressant de combiner les résultats de plusieurs fichiers mesurés individuellement.

Cette attaque peut être menée dans différentes situations :

— Un site web qui veut en savoir davantage sur ses visiteurs.

— Une régie publicitaire pourrait inclure le code de mesure dans les bannières qu'elle distribue afin de faire des statistiques sur les sites web consultés par les visiteurs. Il est même possible de distribuer un code différent pour les catégoriser.

— L'attaquant pourrait créer un site web de telle façon à ce qu'il apparaisse en tête des moteurs de recherche dans le but de faire des statistiques sur les personnes intéressées par un sujet particulier.

— L'attaquant pourrait envoyer un mail contenant le code HTML à sa victime. En le faisant ressembler à du spam, la victime ne remarquerait rien d'anormal et la mesure serait effectuée.

3.3 Pixels espions

3.4 JavaScript

3.4.1 Régies publicitaires sur Internet

3.4.2 Modules des réseaux sociaux

Facebook et son bouton « Like »

Google et son bouton « +1 »

3.4.3 Outils destinés aux webmasters

Google Analytics

3.5 Autres moyens de nous tracer

3.5.1 Flash

3.5.2 Java

3.5.3 Omniprésence des navigateurs dans les applications

3.5.4 Persistance des données utilisateur dans Microsoft Internet Explorer

Chapitre 4

Analyse des principaux sites web

4.1 Objectif de l'analyse

4.2 Description de l'outil implémenté

4.3 Résultats

Chapitre 5

Moyens de défense

5.1 Plugins anonymisants des navigateurs

5.1.1 Ghostery

5.1.2 [...]

5.2 Modes de navigation privée

5.3 Do Not Track

5.4 Utilisation d'un proxy / VPN

5.5 Résultats

Chapitre 6

Conclusion

Bibliographie

[1] Wikipédia - Révélations d'Edward Snowden. `http://fr.wikipedia.org/wiki/R%C3%A9v%C3%A9lations_d'Edward_Snowden`.

[2] Ariane Krol et Jacques Nantel. Pêcher le client dans une baignoire. *Manière de voir*, (133) :21–23, 2014.

[3] Alexa top sites. `http://www.alexa.com/topsites`.

[4] Olivier Bonaventure. Computer Networking : Principles, Protocols and Practice. `http://inl.info.ucl.ac.be/cnp3`.

[5] David M. Kristol. HTTP Cookies : Standards, Privacy, and Politics. *ACM Trans. Internet Technol.*, 1(2) :151–198, November 2001.

[6] D. Kristol and L. Montulli. HTTP State Management Mechanism. IETF - RFC 2109, February 1997.

[7] D. Kristol and L. Montulli. HTTP State Management Mechanism. IETF - RFC 2965, October 2000.

[8] A. Barth. HTTP State Management Mechanism. IETF - RFC 6265, April 2001.

[9] Wikipedia - Web cache. `http://en.wikipedia.org/wiki/Web_cache`.

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. IETF - RFC 2616, June 1999.

[11] Collin Jackson, Andrew Bortz, Dan Boneh, and John C. Mitchell. Protecting Browser State from Web Privacy Attacks. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 737–744, New York, NY, USA, 2006. ACM.

[12] Edward W. Felten and Michael A. Schneider. Timing Attacks on Web Privacy. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, CCS '00, pages 25–32, New York, NY, USA, 2000. ACM.