Université catholique de Louvain

Travail de fin d'études

Analyse du traçage des utilisateurs via leur navigateur Web

Antoine Marchal

Promoteur :
Olivier Bonaventure
Co-promoteur :
Pierre Reinbold
Lecteur :
Xavier Carpent

Mémoire présenté en vue de
l'obtention du grade de
Master [120] en Sciences Informatiques
(option *sécurité & réseaux informatiques*)

Louvain-la-Neuve, Belgique
Année académique 2013-2014

Je souhaiterais remercier
Pierre Reinbold et Olivier Bonaventure
de m'avoir proposé ce sujet et pour leur support
lors de la réalisation de ce mémoire de fin d'études.

C'est un domaine que je ne connaissais pas réellement
avant de l'aborder. Il s'est révélé très intéressant.

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 d'Edward Snowden [1], un informaticien américain et ancien employé de la CIA et de la NSA, ont permis de mettre en lumière un système élaboré destiné à surveiller les utilisateurs d'Internet.

Ce travail se focalise sur 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 consultations et 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 constitué de personnes dont on connaît le profil sociodémographique, 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 moteurs 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 générer plusieurs inconvénients qui nous touchent directement. Il est donc intéressant d'analyser cette surveillance et déterminer par quels moyens elle s'opère. 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 La vie privée

Le respect de la vie privée est intimement lié à l'évolution de la surveillance. Il y a toujours eu des conflits entre les personnes qui estiment que tout le monde doit pouvoir utiliser et partager une technologie permettant de sécuriser ses communications et les gouvernements qui souhaiteraient garder la possibilité d'écouter ces communications. Prenons l'exemple du gouvernement américain qui avait peur de passer d'un monde où il avait un contrôle total à un monde où il n'en avait plus. En conséquence, il a considéré l'export de la cryptographie hors des USA équivalent à l'export de munitions. Ainsi, afin de dévoiler les sources de PGP (Pretty Good Privacy, un logiciel de chiffrement et de déchiffrement cryptographique) au monde entier, son auteur a usé d'une astuce en publiant un livre qui en contient le code source complet (l'export de livres est protégé par le Premier Amendement) [3].

Les programmes de surveillance du passé peuvent être considérés comme de la "surveillance de proximité", où un gouvernement tentait d'utiliser la technologie afin de surveiller les communications lui-même. Les programmes actuels marquent une transition vers la "surveillance oblique" dans laquelle un gouvernement va plus souvent se rendre aux endroits où les informations sont accumulées (fournisseurs d'adresses e-mail, moteurs de recherche, réseaux sociaux et télécoms) [4].

En 2001, John Poindexter, alors qu'il occupait un poste officiel au sein du Département de la Défense des États-Unis, a voulu mettre en place le programme *Total Information Awareness* qui consistait en un système de data mining. Celui-ci devait enregistrer tous les e-mails, tout le trafic web, tout l'historique des cartes de crédit et tous les dossiers médicaux. Le but était ensuite de développer une technologie afin d'extraire les données dont on avait besoin.

Lors de sa conférence à la DEF CON 18, Moxie Marlinspike [1] a tourné en ridicule le logo du programme (Figure 1.1) en expliquant qu'il était déconseillé d'utiliser un logo qui inspirait la peur. D'ailleurs, le projet a été annulé car il a reçu beaucoup de contestations. Cependant, lorsque l'on regarde ce que Google fait aujourd'hui, c'est exactement la même chose, voire pire. Or, personne ne proteste et de surcroît, tout le monde utilise ses services !

— Les emails sont enregistrés par Gmail.
— Le trafic web est enregistré par Google Analytics.
— L'historique des cartes de crédit est enregistré par Google Checkout.
— Les dossiers médicaux sont enregistrés par Google Health.
— L'historique des positions GPS est enregistré par Android.
— ...

De plus, on sait que Google a la capacité d'extraire de manière efficace les informations qu'il désire afin de les monétiser avec des publicités. Lorsque le CEO de Google, Eric Schmidt, déclare : "If you have something that you don't want anyone to know, maybe you shouldn't be doing it in the first place" [6], cela fait froid dans le dos.

FIGURE 1.1 – Le logo officiel du programme *Total Information Awareness*.

Dans sa conférence, Moxie Marlinspike utilise un autre exemple : personne n'accepterait de porter un tracker monitoré par l'État mais tout le monde possède et porte un GSM en permanence. Or, celui-ci envoie en temps réel sa position à l'opérateur qui est obligé de délivrer cette information si on la lui demande. Pour lui, il y a cependant une différence essentielle : le choix. Nous choisissons de posséder un GSM afin de communiquer mais nous refuserions un dispositif de traçage s'il nous était imposé.

1. Moxie Marlinspike est le pseudonyme d'un chercheur en sécurité informatique qui est engagé dans le respect de la vie privée [5].

Maintenant, passons au sujet de la vie privée.
D'après Moglen [2] [7], le concept de vie privée englobe 3 éléments.

- La confidentialité : la capacité de garder des messages privés c'est-à-dire que leur contenu n'est connu que de ceux à qui ils sont destinés.
- L'anonymat : la capacité d'envoyer des messages sans savoir de qui ils proviennent ni à qui ils sont destinés, même si leur contenu est ouvert.
- L'indépendance : la capacité de prendre des décisions librement, sans que la confidentialité ou l'anonymat ne soient violés.

Pour lui, si l'un de ces éléments n'est pas respecté, on ne peut pas avoir de gouvernement démocratique car la vie privée en est une condition nécessaire.

Il explique également qu'on ne devrait pas voir la vie privée comme une transaction. En effet, ceux qui veulent profiter de nos informations souhaitent définir la vie privée comme un concept que l'on peut négocier. Ainsi, ils nous offrent un service gratuit comme l'accès à une messagerie mais en échange, ils lisent nos mails. Pour eux, il s'agirait seulement d'une transaction entre deux parties. Cependant, si on y réfléchit, ce n'est pas une transaction quelconque car tous ceux qui nous écrivent sont aussi impliqués dans cet accord, qui était supposé être bilatéral.

Rien n'est gratuit dans la vie et cela l'est également sur Internet. En réalité, le prix à payer est de dévoiler des informations personnelles qui peuvent potentiellement être exploitées afin de rapporter de l'argent.

Les exemples de l'introduction montrent déjà clairement que la surveillance des sites web peut nous coûter de l'argent en nous incitant à consommer plus. Mais les risques de cette surveillance peuvent aller bien au-delà. En effet, la liste des sites web visités par une personne peut en révéler beaucoup sur sa situation familiale, financière ou médicale.

Imaginons une situation où le directeur des ressources humaines d'une entreprise reçoit des CV. Il décide de trouver davantage d'informations au sujet des candidats et pour cela, tape leur nom dans un moteur de recherche. Il pourrait tomber sur la photo d'une jeune femme portant un bébé dans ses bras ou sur la photo d'une jeune femme seule. Pensez-vous qu'elles aient la même chance d'être rappelées pour un entretien ?

2. Eben Moglen est professeur de droit et d'histoire du droit à l'Université Columbia. Il est le président du Software Freedom Law Center, une organisation qui procure assistance et défense juridique aux développeurs de logiciels libres et open-source.

Lors d'une expérience menée avec son équipe [8], Alessandro Acquisti [3] a créé des profils Facebook en modifiant certaines de leurs caractéristiques. Il a ensuite envoyé des candidatures à différentes entreprises américaines et a analysé leur comportement face aux profils associés aux candidatures. Les résultats de l'expérience ont montré que les entreprises agissaient en fonction des informations qu'elles avaient trouvées sur les réseaux sociaux.

Face à la montée de la surveillance, certaines personnes émettent le désir de préserver leur vie privée. On leur rétorque souvent "Si vous ne faites rien de mal, qu'avez-vous à cacher ?". D'après Bruce Schneier [4] [9], la vie privée n'a rien à voir avec le fait d'avoir quelque chose de négatif à cacher, c'est un besoin humain essentiel. Nous faisons énormément de choses en privé qui ne sont pas considérées comme illégales mais que nous préférons garder pour nous. Mettre des rideaux à ses fenêtres n'est pas criminel, cela traduit juste notre envie de garder notre intimité, vie privée. Ce qui l'ennuie c'est qu'on présuppose que la vie privée sert à dissimuler des mauvaises actions alors que ce n'est pas le cas. Il affirme même que le droit à une vie privée est un droit de l'être humain. D'ailleurs, lorsque la Constitution américaine fut rédigée, le principe de vie privée n'y a pas été inscrit car il était considéré comme inhérent aux êtres. A l'époque, on ne surveillait que les criminels, pas les honnêtes citoyens.

Lorsqu'une personne se sent surveillée, elle n'agit plus de la même façon. Cela vous est sûrement déjà arrivé de cesser votre conversation si vous vous sentez observés, par peur que vos mots ne soient sortis de leur contexte. Une surveillance généralisée a le même effet mais sur un ensemble de personnes beaucoup plus étendu.

Est-ce que les personnes déclarant ne rien avoir à cacher resteraient du même avis si leur gouvernement, avec l'aide de toutes les informations qu'il possède grâce à la surveillance, décidait que leurs actions étaient suspectes et que des sanctions étaient appliquées arbitrairement ?
Une simple question pourrait également les raisonner : donneriez-vous votre adresse, numéro de téléphone, numéro de carte de crédit et d'autres données personnelles à un inconnu dans la rue ?

3. Alessandro Acquisti est un économiste comportemental à l'Université Carnegie Mellon et il étudie l'économie comportementale de la vie privée.

4. Bruce Schneier est un cryptologue, spécialiste en sécurité informatique et écrivain américain.

1.3 Méthodologie utilisée

Afin de répondre à la problématique de la surveillance sur Internet, nous voulons tout d'abord comprendre comment Internet fonctionne et en particulier, certains de ses mécanismes tels que les cookies.

Ensuite, nous voulons 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 visités au monde.

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

Les sources ainsi que l'exécutable de l'outil développé sont disponibles publiquement sur `https://github.com/manto235/thesis_final`. Une version électronique du mémoire y est également présente.

Chapitre 2

Protocoles et mécanismes d'Internet

2.1 Présentation du protocole HTTP

HTTP (HyperText Transport Protocol) [10] 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. Il est décrit par la RFC 2616 [11].

Les requêtes HTTP se composent de trois parties :

1. Une ligne de requête (qui contient notamment une URI)
2. Des entêtes (qui contiennent des paramètres 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" présent dans ce lien. Le navigateur envoie alors une requête à laquelle le serveur répond puis le navigateur se déconnecte du serveur. 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 [10].

Les réponses HTTP se composent également de trois parties :

1. Une ligne de statut (qui indique si la requête est réussie ou pas)
2. Des entêtes (qui contiennent des informations sur la réponse)
3. Le corps de la réponse

Étant donné qu'HTTP est sans état, il fallait trouver une solution afin d'avoir la possibilité de garder une certaine quantité d'informations entre des requêtes successives. En effet, si le serveur est dans l'incapacité de se souvenir de ce que le client a fait auparavant, il est par exemple impossible de développer un site de e-commerce car celui-ci ne sauvegarderait pas la liste des articles du panier virtuel. Plusieurs solutions ont été imaginées [10].

Une première solution consistait à forcer l'authentification du client (comme pour FTP) mais ce n'est pas toujours nécessaire ou applicable sur tous les sites web.

Une seconde solution était d'utiliser les différents types d'entêtes *Accept-**. Par exemple, un client aurait pu utiliser l'entête *Accept-Language* pour indiquer la langue dans laquelle il voulait visiter le site. Cependant, cela fournit des possibilités assez limitées car cet entête est réglé par le navigateur et l'utilisateur aurait été dans l'impossibilité d'indiquer une langue différente pour chaque site visité sans devoir effectuer une manipulation conséquente.

Une autre solution, qui est la plus largement adoptée, est l'utilisation d'un cookie HTTP. Elle est expliquée plus en détail dans la section suivante.

Des exemples de requête et réponse HTTP sont disponibles dans la section 2.4.

2.2 Cookies

Une première description informelle sur les cookies a été publiée sur le site web de Netscape Communications. 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 [12]. C'est ainsi que la première RFC (RFC 2109) sur les cookies est parue en février 1997 [13], rendue obsolète en octobre 2000 par la RFC 2965 [14]. Celle-ci a également été remplacée, par la RFC 6265 en avril 2011 [15], qui est toujours d'application en ce jour et constitue la référence en ce qui concerne les cookies. Il est intéressant de noter qu'une section dédiée à la vie privée (section 7) est présente dans cette dernière RFC, ce qui constitue une nouveauté.

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 entêtes des requêtes et des réponses HTTP [15].

Notez que certains navigateurs actuels enregistrent désormais ces cookies dans une base de données.

C'est notamment le cas pour Firefox 3 qui enregistre ses cookies dans une base de données SQLite.

Dans sa réponse à une requête, un serveur peut envoyer des informations arbitraires (le cookie) dans un entête *Set-Cookie*. Le client a le choix d'ignorer cet entête. S'il l'accepte, l'information présente dans l'entête 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,... Le serveur peut ajouter des attributs afin de configurer les cookies (sections 5.2.1 à 5.2.6 de la RFC 6265) [15] :

— Expires et Max-Age : date d'expiration
— Domain : domaine
— Path : chemin
— Secure et HttpOnly : type de connexion

Grâce aux attributs *Domain* et *Path*, le serveur peut décider à qui le client va renvoyer cette information. Par exemple, avec un attribut *Domain=exemple.com* et un attribut *Path=/*, le serveur demande au client de renvoyer le cookie dans chaque requête vers tout sous-domaine et chemin de *exemple.com*.

Lorsqu'une requête est envoyée à un serveur et si l'URI présente dans cette requête correspond aux attributs *Domain* et *Path* d'un cookie enregistré chez le client, celui-ci l'ajoute à sa requête. Habituellement, le client est coopératif et renvoie l'information du cookie qui est stocké par le navigateur. Ainsi, dans chaque requête ultérieure qu'il fait vers le serveur, le client indique cette information dans un entête *Cookie*. Le serveur peut choisir de renvoyer un nouveau cookie dans ses réponses, ce qui remplacera automatiquement l'ancien cookie. Il peut également supprimer un cookie en renvoyant un entête *Set-Cookie* avec une date d'expiration dans le passé. Le cookie sera supprimé seulement si les attributs *Path* et *Domain* dans l'entête *Set-Cookie* correspondent aux valeurs utilisées lorsque le cookie a été créé.

Il y a un contrat implicite entre le client et le serveur : le serveur compte sur le client pour lui retourner son état lors de la prochaine requête. Un cookie est donc une donnée que le serveur et le client se renvoient l'un à l'autre. La quantité d'informations est généralement petite et 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.

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. Le principe de cache sur le protocole HTTP est défini dans la RFC 2616 [11] (sections 13 et 14). Les mécanismes de cache peuvent être implémentés 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

Lorsqu'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, les administrateurs d'un site web peuvent paramétrer la mise en cache de certains éléments grâce à l'entête *Cache-Control*. Celui-ci est utilisé pour spécifier des directives qui doivent être respectées par tous les mécanismes de cache le long de la chaîne requête-réponse. Grâce à ces directives, il est possible de spécifier explicitement comment chaque fichier doit être traité vis-à-vis de la mise en cache [11].

Ces directives de réponse HTTP permettent notamment d'autoriser (ou d'empêcher) la mise en cache d'un fichier, d'empêcher la mise en cache partagé ou encore, de préciser une date d'expiration à partir de laquelle le fichier présent en cache devra être revalidé.

2.4 Exemple de requête et réponse HTTP

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
```

Note : aucune visite vers ce site n'a été faite auparavant.

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
```

A la ligne 9, on peut voir un entête *Set-Cookie*. Un cookie dont le nom est *"NewLastSeen"* est donc créé avec la valeur "8407". Lors d'une prochaine visite sur le site, ce cookie sera alors renvoyé automatiquement dans la requête HTTP par le navigateur. On peut également voir un entête *Cache-Control* à la ligne 7.

Voici maintenant une seconde requête vers le même site lors de la même session. On voit que le navigateur envoie le cookie enregistré précédemment car la requête HTTP contient maintenant un entête *Cookie* visible à la ligne 7 qui n'était pas présent dans la première requête :

```
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.

Dans la seconde requête, le navigateur envoie 5 cookies alors qu'un seul a été enregistré via la réponse HTTP analysée auparavant.
Ceci s'explique par le fait que les cookies *__utmX* proviennent de Google Analytics [16] et servent à analyser la navigation des visiteurs du site.

Il n'est pas nécessaire d'analyser les requêtes échangées pour voir les cookies envoyés. Dans les préférences du navigateur, on peut remarquer que 4 cookies sont présents pour l'hôte *distrowatch.com* (Figure 2.1). Le cookie *__utmc* n'apparaît plus car il a été supprimé à la fin de la session de navigation (suite à sa date d'expiration, voir section 3.5.3) et que la capture d'écran a été effectuée lors d'une session ultérieure.

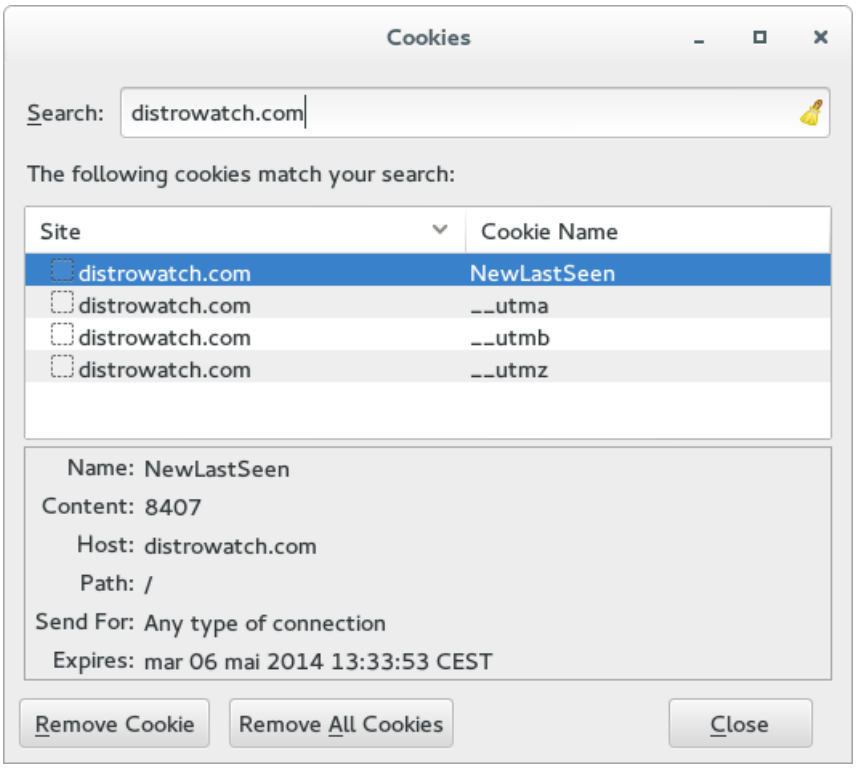

FIGURE 2.1 – Les cookies enregistrés pour l'hôte *distrowatch.com* dans Firefox.

2.5 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 fortement 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 [11]). En contrepartie, cela permet aux serveurs en question d'enregistrer des données sur le client suite aux requêtes qu'il effectue.

2.6 Principe de même origine

Le principe de même origine (*same-origin principle*) peut être déclaré comme suit : "*seul le site qui enregistre des données dans le navigateur peut lire ou modifier ces données plus tard*" [17]. Son but est d'isoler des sites en respectant leur capacité à lire ou modifier l'état du navigateur. Cela permet à un utilisateur de naviguer comme si chaque site et chaque session étaient complètement indépendants.

Lors du développement de Netscape Navigator 2, une décision importante a été prise concernant le principe de même origine. Celui-ci interdit des sites web de domaines différents d'interagir entre eux au niveau du cache du navigateur, sauf dans des cas précis. Plus concrètement, son but est de permettre aux cookies et aux codes JavaScript de sites web de confiance variable de coexister silencieusement dans le navigateur de l'utilisateur, sans interférer mutuellement. Ce principe est désormais implémenté dans les différents navigateurs mais il n'est pas appliqué de la même manière. En effet, le principe n'a ni été déclaré de manière générale ni appliqué de manière uniforme aux multiples façons dont un site web peut sauvegarder ou récupérer des données sur la machine d'un utilisateur.

Il n'est pas facile de déterminer quelles actions devraient être sujettes à un contrôle de sécurité [18]. Il semble clair que certaines interactions, comme suivre un lien, devraient être permises sans restrictions spéciales étant donné qu'elles sont essentielles au fonctionnement de l'environnement Web. En effet, c'est le principe même sur lequel repose le Web avec les liens hypertextes. A l'opposé, d'autres interactions, comme modifier le contenu d'une page dans une autre fenêtre, devraient subir un contrôle de sécurité.

L'origine telle que définie selon le principe de même origine est le triplet protocole-hôte-port. Différents exemples issus du site de Mozilla[1] sont disponibles dans la Figure 2.2.

La simplicité du principe de même origine est à la fois une aubaine et une malédiction [18]. Son mécanisme est facile à comprendre et pas trop compliqué à im-

1. Mozilla est le développeur du navigateur Firefox.

URL	Résultat	Raison
http ://store.company.com/dir2/other.html	Succès	/
http ://store.company.com/dir/inner/another.html	Succès	/
https ://store.company.com/secure.html	Échec	Protocole différent
http ://store.company.com :81/dir/etc.html	Échec	Port différent
http ://news.company.com/dir/other.html	Échec	Hôte différent

FIGURE 2.2 – Différents exemples d'URL et leur comparaison par rapport à *http ://store.company.com/dir/page.html* selon le principe de même origine [19].

plémenter mais son inflexibilité peut être une limite pour les développeurs. Il est parfois trop vaste, rendant impossible l'isolation de pages à des utilisateurs différents (à moins de faire un sous-domaine pour chaque utilisateur) et il peut rendre le partage d'informations difficile pour des sites coopérant de façon légitime (entre *login.exemple.com* et *paiements.exemple.com* par exemple).

2.7 Conclusion

Comme nous avons pu le voir dans ce chapitre, le protocole HTTP a été amélioré au fil du temps grâce à de nouvelles fonctionnalités. La plus importante d'entre elles est la possibilité pour le serveur de garder de l'information entre des requêtes HTTP successives issues d'un client. Sans cette fonctionnalité, le Web tel qu'on le connaît aujourd'hui n'existerait pas. Cependant, lorsque ces mécanismes ont été mis en place, la situation était différente et les aspects liés à la vie privée n'étaient pas encore d'actualité. Il en résulte que ces mécanismes ne proposent pas de solution adéquate afin d'éviter certaines attaques et c'est la raison pour laquelle le respect de la vie privée est un thème qui a attiré davantage l'attention au fil du temps.

Chapitre 3

Moyens d'identification

Dans ce chapitre, nous allons nous intéresser aux techniques qui permettent de tracer les utilisateurs. Plus précisément, nous allons voir comment ces techniques utilisent ou détournent les mécanismes assurant le fonctionnement du Web.

3.1 Imperfections du principe de même origine

L'échec de la bonne adaptation de ce principe est une source importante de fuites de données et d'attaques. Les violations du principe de même origine peuvent être dues à un filtrage de script insuffisant du côté des applications Web du serveur ou à des défauts dans les mécanismes d'isolation des domaines au sein du navigateur [20].

Les défauts de filtrage des scripts au niveau des serveurs sont communément appelés failles XSS (*cross-site scripting*). En exploitant ces failles, des scripts malveillants peuvent passer à travers le filtrage et être exécutés dans le même contexte de sécurité que les applications Web authentiques.

Au niveau des navigateurs Web, les violations du principe de même origine sont dues à une mauvaise isolation des contenus des différents domaines. Certains domaines peuvent alors accéder aux informations appartenant à d'autres domaines alors qu'ils ne devraient pas y avoir accès.

Lorsqu'un attaquant désire compromettre une application, il est parfois plus facile d'attaquer les utilisateurs à travers leur navigateur Web que le serveur lui-même [21]. Les navigateurs sont défendus par le principe de même origine mais la présence de failles dans le site Web ou dans le serveur peuvent permettre à l'attaquant de passer outre ce principe. La principale vulnérabilité des sites sont les attaques XSS. Celles-ci permettent à un attaquant de placer son code au sein des pages d'une application Web vulnérable. Lorsqu'un visiteur se rend sur l'une de ces

pages, le code de l'attaquant est chargé avec les autres éléments de la page. Ce script malicieux peut alors être en mesure de lire le cookie de l'utilisateur ou récupérer des valeurs affichées uniquement à l'utilisateur en utilisant du JavaScript. Ensuite, une requête HTTP contenant les données volées serait utilisée afin de les envoyer au serveur de l'attaquant. Celui-ci serait ensuite capable de se faire passer pour l'utilisateur légitime sur le site ou d'utiliser les informations reçues à mauvais escient (il pourrait par exemple s'agir de données bancaires), voir Figure 3.1.

```
1  <script>
2    // Vol du numero de client
3    var numeroDuClient = document.getElementById('numClient').
       innerHTML;
4    // Vol du cookie
5    var cookieDuClient = document.cookie;
6    // Requete contenant les informations volees :
7    var requete = 'http://serveur-du-voleur.be/vol?client=' +
       numeroDuClient + '&cookie=' + cookieDuClient;
8    // Envoi de la requete avec une image
9    document.write("<img src='" + requete + "'/>");
10 </script>
```

FIGURE 3.1 – Un exemple d'attaque XSS pour voler des informations.

Quand le client va se rendre sur la page, son navigateur va automatiquement faire une requête vers le serveur du voleur. Son navigateur l'avertira seulement que l'image n'a pas été trouvée mais les informations auront été néanmoins transmises au voleur. Cette attaque nécessite un minimum de préparation car le voleur doit savoir quel est le nom ou l'ID de l'élément HTML qui affiche le numéro de client (dans l'exemple, "numClient"). Ceci est un premier exemple de violation du principe de même origine réalisé avec l'aide d'une faille XSS.

Il existe plusieurs moyens de contourner le principe de même origine. Le but de ce chapitre n'étant pas d'en faire une liste exhaustive, seuls quelques moyens seront expliqués. Il faut également préciser que même si le principe était correctement implémenté au sein d'un navigateur, il n'empêcherait pas les violations de vie privée face à des sites coopérants. Ceci s'explique par le fait qu'il existe une multitude de techniques simples allant des redirections jusqu'aux liens inter-sites pouvant être utilisées dans le but de transmettre des données entre des sites. Avec de tels échanges, les sites coopérants sont dans la capacité de constituer un profil inter-domaine des activités de leurs visiteurs.

3.2 Cookies

En général, on utilise deux types de classification pour les cookies [22].

Le premier se base sur l'origine et la destination : les cookies sont classifiés en tant que cookies "first-party" ou "third-party". Les cookies "first-party" sont issus du domaine que l'utilisateur est en train de visiter, on les appelle cookies d'origine ou cookies de domaine. Les cookies "third-party" sont créés par un site autre que celui qui est visité par l'utilisateur, on les appelle cookies tiers.
Les cookies tiers sont placés suite à des réponses HTTP venant de sites tiers. Un site *alpha.com* ne peut pas créer de cookie au nom du site *beta.com* (voir RFC 6265 [15], section 4.1.2.3), cela provoquerait des problèmes de sécurité évidents. Pour les mêmes raisons, les cookies ayant un suffixe public [1] dans l'attribut *Domain* sont rejetés.

La seconde méthode de classification se base sur la durée de vie du cookie : les cookies sont alors classifiés en cookies de session ou cookies persistants. Les cookies de session sont stockés dans la mémoire vive de l'ordinateur et supprimés à la fermeture du navigateur. A l'opposé, les cookies persistants sont stockés sur le disque dur de l'ordinateur et supprimés soit lors de leur expiration, soit manuellement par l'utilisateur.

Les cookies tiers, qu'ils soient de session ou persistants, n'apportent quasiment aucun bénéfice à l'utilisateur. Ils sont d'ailleurs reconnus comme une menace pour la vie privée et les navigateurs proposent généralement une option pour les désactiver.

Les cookies d'origine de type session ne posent généralement pas vraiment de problèmes relatifs à la vie privée étant donné leur faible durée de vie. En théorie, ceci est valable car l'utilisateur ferme les fenêtres de son navigateur régulièrement. Par contre, un utilisateur qui laisse constamment l'onglet d'un site ouvert pourrait se faire tracer par le site en question.

A l'opposé, les cookies d'origine persistants peuvent poser des problèmes divers. En effet, les cookies persistants sont une arme à double tranchant : ils peuvent jouer un rôle utile en permettant la personnalisation et l'authentification sur les sites mais ils peuvent également jouer un rôle plus dangereux qui amène des risques au niveau de la vie privée et de la sécurité. Ces risques reposent sur deux aspects : le premier, et celui qui nous intéresse principalement ici, est que les cookies persistants permettent de tracer l'activité de l'utilisateur dans le temps. En effet, pour chaque page visitée, le client envoie une requête au serveur, celui-ci est donc capable de suivre l'utilisateur lors de la visite du site. Le deuxième risque est que les cookies persistants

1. Un suffixe public est un domaine qui est contrôlé par un registre public comme "com", "co.uk" ou "pvt.k12.wy.us" [15]. Une liste est maintenue par le projet Mozilla : `http://publicsuffix.org/`.

peuvent être volés ou manipulés par des attaques de deux types : les attaques XSS (elles exploitent les vulnérabilités des applications Web, voir la Figure 3.1 de la section 3.1) et les attaques qui exploitent les vulnérabilités des navigateurs Web (via notamment le contournement du principe de même origine).

Une étude menée sur plus de 5000 sites à propos de l'utilisation des cookies [22] a montré que les cookies d'origine persistants sont largement utilisés et que plus de 60% sont réglés pour expirer plus d'un an après leur création.

Désactiver tous les cookies tiers et garder les cookies d'origine de type session est supporté par la majorité des navigateurs actuels. Le problème se situe au niveau des cookies d'origine persistants car on ne sait pas comment les gérer et déterminer s'ils sont utiles ou néfastes pour l'utilisateur.

3.3 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.

3.3.1 Tracking par mesure du temps

Une méthode d'exploitation du cache à des fins de surveillance est détaillé par Felten et Schneider [23]. 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 précisément, 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.

Le plus fiable et facile est d'utiliser un applet Java ou un code JavaScript qui va mesurer le temps de chargement du fichier à partir de son URL. Même si l'utilisateur a désactivé l'exécution de Java et de JavaScript, il est possible d'obtenir une mesure suffisamment 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 raffiner 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.

D'une manière semblable, ce type d'attaque peut également être réalisé sur le cache du DNS. Felten et Schneider ont vérifié la précision des tests avec l'aide de 3 serveurs distincts situés à des distances différentes.

Plus le serveur est situé à une grande distance, plus la requête DNS prend de temps. Les tests ont montré que la précision des résultats pouvait être inférieure car la pénalité suite à des défauts de cache était très petite. Il n'était donc pas possible de discriminer les échecs des succès de cache. Cependant, sur Internet, la pénalité en cas de défaut est suffisamment grande pour assurer une bonne distinction. Dans ce cas-là, les tests montrent une très bonne précision.

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. Bien entendu, cette attaque résulte d'une collaboration entre le site et la régie publicitaire.

— 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. Il faut noter que cette situation reste théorique et a peu de chances de fonctionner.

— 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.

3.3.2 Stockage local HTML5

Une nouveauté d'HTML5 est la possibilité de stocker localement des données pour un usage hors-ligne [24]. Le stockage local pourrait servir de "super-cookie" car il permet de stocker jusqu'à 5MB de données. De plus, ces données sont dites persistantes car elles doivent être supprimées par l'utilisateur ou le site Web.

Dans une expérience portant à l'origine sur le traçage via les cookies Flash [25] (voir section 3.6), les chercheurs ont ajouté cette nouvelle dimension apportée par l'HTML5. L'expérience réalisée une première fois en 2009 portait sur les cookies Flash et les cookies HTTP. Une expérience similaire a été réalisée à nouveau en 2011 en apportant également des informations relatives au stockage permis par l'HTML5.

Les analyses ont été réalisées sur le TOP 100 donné par Quantcast [2] en juillet 2011. Il en ressort que 17 sites sur les 100 ont utilisé le stockage local HTML5 pour un total de 60 paires clé-valeur. Les chercheurs ont trouvé des correspondances entre le stockage local HTML5 et les cookies HTTP dans différents cas (*twitter.com*, *tmz.com*, *squidoo.com*, *nytimes.com*, *hulu.com*, *foxnews.com* et *cnn.com*). Dans la plupart de ces cas, la valeur commune aux deux éléments était issue d'un service tiers (comme *meebo.com*, *kissanalytics.com* et *polldaddy.com*).

3.3.3 ETags

Le champ d'entête *ETag* est défini dans la RFC 2616 [11]. Il permet de déterminer si l'élément présent en cache dans le navigateur correspond à celui présent sur le serveur. Il est choisi arbitrairement par le serveur et est placé dans les réponses

2. Quantcast est une entreprise spécialisée dans la mesure d'audience et la publicité.

HTTP. Lorsqu'un client fait une requête vers un serveur et qu'il possède une copie du fichier en cache avec son *ETag*, le serveur va comparer les deux *ETags*. Cela permet d'économiser de la bande passante car le serveur n'envoie pas l'élément si les deux *ETags* correspondent. Un exemple est disponible à la Figure 3.2 [3].

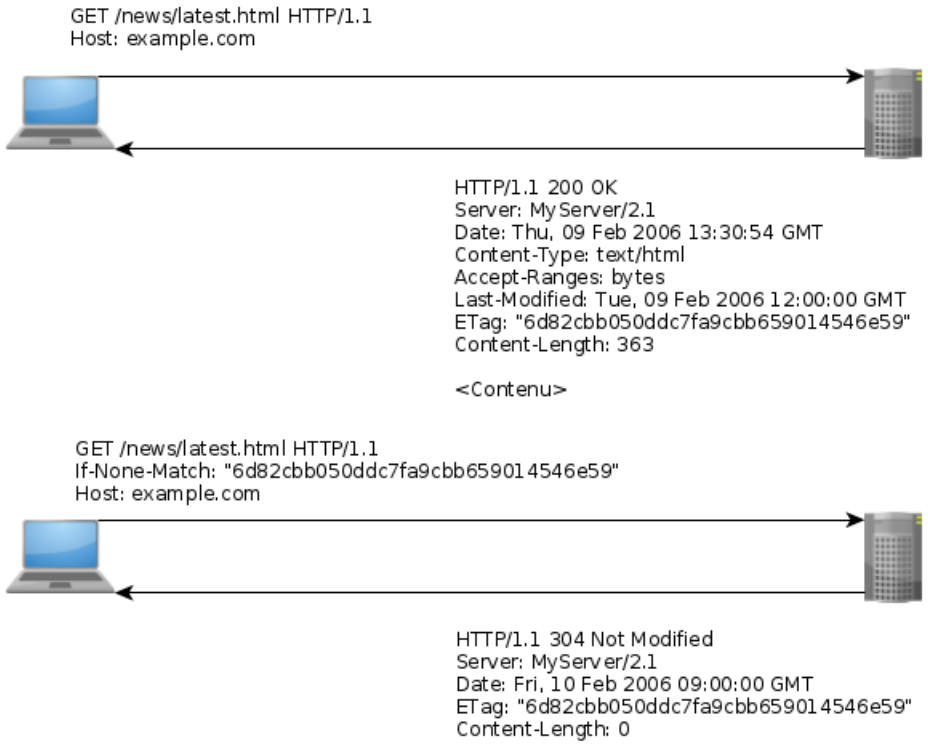

FIGURE 3.2 – Un exemple de requêtes et réponses HTTP utilisant l'*ETag*.

Dans l'expérience mentionnée dans la sous-section 3.3.2 et expliquée dans la section 3.6, les chercheurs ont trouvé un cas de réapparition de cookie HTTP (un cookie HTTP a été supprimé par l'utilisateur puis un cookie identique a été recréé grâce au stockage local HTML5) sur *hulu.com* à travers un service hébergé sur *kissmetrics.com*. La technique a utilisé les ETags afin de sauvegarder les valeurs.

D'après les chercheurs, le tracking via l'entête ETag et le fait de pouvoir recréer des cookies à l'identique est particulièrement problématique car cette technique génère des valeurs de tracking uniques même si l'utilisateur bloque les cookies HTTP et Flash. En effet, afin d'empêcher ce traçage, l'utilisateur devrait vider son cache entre chaque site visité. De plus, en mode de navigation privée, l'utilisateur peut quand même être suivi durant une même session [25].

3. Exemple basé sur `http://www.w3.org/2005/MWI/BPWG/techs/CachingWithETag.html`

3.4 Pixels espions

Les pixels espions sont des images de taille minime (généralement, de 1 pixel de haut sur 1 pixel de large) et sont destinés à effectuer une requête HTTP vers un serveur sans que le visiteur de la page ne s'en aperçoive. L'administrateur du site hébergeant ce pixel espion peut dès lors analyser les logs de son serveur Web afin d'obtenir des informations sur l'utilisateur. En plaçant un pixel espion spécifique sur chaque site, il est possible de distinguer sans problème les requêtes provenant des différents sites et ainsi déterminer quel site a été visité par chaque utilisateur. En analysant la requête envoyée par le client, le serveur peut connaître plusieurs informations : la date et l'heure à laquelle le visiteur s'est rendu sur la page, son "User-Agent" (le navigateur utilisé, sa version, le système d'exploitation,...), la langue, les cookies et toutes les informations disponibles par défaut dans une requête HTTP. De plus, l'URI présente dans celle-ci peut contenir des informations additionnelles placées dans la chaîne de requête de l'URI (après le point d'interrogation), comme cela est illustré dans la Figure 3.1.

Les pixels espions sont généralement utilisés pour tracker les visites. Par exemple, un site commercial pourrait placer un pixel de tracking sur sa page de confirmation d'achat. Cela enverrait alors une requête vers un serveur qui analyse les requêtes reçues et le parcours de l'acheteur pourrait être retracé avec l'information contenue dans le cookie. Par exemple, un cookie pourrait contenir le nom de la campagne de publicité qui a amené le visiteur sur le site. Les statistiques reçues permettraient alors au site de déterminer quelle campagne marketing a le mieux fonctionné.

3.5 JavaScript

JavaScript permet d'exécuter des scripts du côté client. Il a entraîné le déploiement d'applications riches et accessibles simplement via un navigateur Web. Les possibilités sont multiples : il est ainsi facile de se divertir grâce à un jeu écrit en JavaScript mais il est également aisé d'en savoir plus sur l'utilisateur via le navigateur qui exécute le script. Ceci est possible grâce à l'accès dont dispose JavaScript sur l'ordinateur du client. En effet, différentes attaques peuvent être perpétrées avec JavaScript afin d'identifier les utilisateurs [26] :

— Le vol de cookies : le script inclus depuis un site tiers peut accéder à toutes les informations présentes sur la page. Lorsque celle-ci fait appel à un script, elle lui donne accès aux cookies, à la barre d'adresses et à l'ensemble des éléments disponibles sur la page. Le script a donc la capacité de lire le contenu des

cookies associés au site et de l'envoyer à un site tiers.

— Le détournement d'adresse : le script, ayant l'accès complet au contenu de la page, peut influencer les valeurs des URL. Le script peut également rediriger le navigateur vers un autre site et le ramener ensuite vers le site d'origine sans que l'utilisateur ne s'en aperçoive (au chargement de la page par exemple).

— L'analyse de l'historique : l'attaque consiste à regarder comment les liens sont affichés par le navigateur. En effet, s'ils ont été visités, les liens s'affichent d'une autre couleur. Avec JavaScript, il suffit alors de créer un lien vers le site que l'on désire cibler dans une partie invisible de la page et utiliser l'interface DOM du navigateur pour regarder comment celui-ci affiche le lien. Cette attaque est possible car dans la plupart des navigateurs, l'accès à un historique de pages visitées, de fichiers cachés et de cache DNS est partagé entre les domaines. Cependant, ces éléments ne sont pas accessibles simplement. Par exemple, il est nécessaire de placer un lien vers un site pour accéder à l'information concernant ce site et pouvoir déterminer s'il a été visité grâce à la couleur du lien affiché.

— Le traçage de comportement : il est possible de déterminer avec précision le comportement de l'utilisateur sur la page qu'il visite. L'élaboration d'une ligne du temps avec les interactions de l'utilisateur (clics, mouvements, défilements, parties de texte surlignées,...) est réalisable avec l'aide de gestionnaires d'événements. Cet ensemble d'interactions peut alors être envoyé à un site tiers afin de calculer des statistiques sur la navigation de l'utilisateur.

Lorsque du code JavaScript est inclus dans une page, il n'est pas soumis au principe de même origine (section 2.6). Ce script a donc accès à l'ensemble des éléments de la page et pourrait potentiellement récupérer des données, voir Figure 3.3. [21].

```
1 <script src="http://exemple.com/script.js />
```

FIGURE 3.3 – Inclusion d'un code JavaScript dans une page.

Une propriété de JavaScript mérite aussi notre attention, il s'agit de *document.domain*. Cette propriété permet à deux sites ayant le même domaine de plus haut niveau de s'accorder sur le fait qu'ils veulent être considérés équivalents. Par exemple, afin de pouvoir s'échanger des données, deux sites *login.exemple.com* et *paiements.exemple.com* pourraient passer outre la vérification habituelle du domaine (les protocoles et les ports doivent correspondre), voir Figure 3.4.

```
1 document.domain = "exemple.com"
```

FIGURE 3.4 – Modification du domaine de la page avec JavaScript.

3.5.1 Régies publicitaires sur Internet

Il semble logique que les régies publicitaires se soient intéressées au tracking des utilisateurs. Grâce à lui, elles sont en mesure de calculer des statistiques diverses sur les intérêts des utilisateurs, ceci ayant pour but de les inciter à consommer davantage en leur proposant par exemple des produits qui ont plus de chances de les intéresser. Le tracking leur permet également de calculer les montants que leurs clients vont payer (pour ceux qui affichent les publicités de leurs produits) ou recevoir (pour ceux qui affichent de la publicité sur leur site). Ainsi, Google possède 2 services : *Google Adsense* qui utilise les contenus des utilisateurs (sites Web, vidéos YouTube,...) comme support d'affichage pour les publicités et *Google Adwords* qui propose à des annonceurs de diffuser leurs publicités.

Les techniques utilisées par les régies publicitaires sont globalement les mêmes que pour les autres entreprises : il faut inclure un code JavaScript qui va s'occuper de faire le tracking des utilisateurs et ce code contient généralement l'identifiant du client de la régie publicitaire. Il faut noter que les régies publicitaires sont parfois prêtes à aller loin pour s'assurer du suivi des utilisateurs. C'est pourquoi certaines n'hésitent pas à user de techniques malicieuses pour recréer des cookies HTTP effacés grâce à Flash ou au stockage local HTML5 (voir les sections correspondantes).

La publicité sur Internet constitue un important modèle économique qui supporte l'accès gratuit à une majorité de contenus. Afin de pouvoir gagner un profit, les éditeurs placent des publicités sur leur site et se font payer par des publicitaires ou réseaux publicitaires. Vu la popularité du Web aujourd'hui, ces derniers veulent que leurs techniques soient de plus en plus sophistiquées. Au lieu d'afficher la même publicité chez tous les utilisateurs, ils souhaitent que leurs publicités soient personnalisées. Afin de parvenir à faire de la publicité ciblée, ils doivent être en mesure de déterminer le profil de chacun et c'est pour cela que le traçage des utilisateurs semble s'intensifier avec le temps [27].

3.5.2 Modules des réseaux sociaux

Les réseaux sociaux proposent généralement de placer un module social qui permet aux visiteurs d'un site d'interagir et de partager avec leurs amis. Bien sûr,

l'intégration d'un tel script permet aux plateformes de réseaux sociaux de suivre la navigation des visiteurs d'un site. Si beaucoup de sites intègrent ces scripts, les réseaux sociaux sont alors en mesure d'obtenir une sorte de carte d'Internet avec les déplacements et actions de chaque utilisateur. En effet, à chaque page contenant un module social, une requête est effectuée vers le serveur du réseau social ; celui-ci peut donc le suivre. De plus, si le visiteur est connecté à son compte sur le réseau social en naviguant sur d'autres sites, la plateforme peut directement faire le lien entre son profil et les sites qu'il visite.

Les principaux réseaux sociaux proposent généralement les mêmes fonctionnalités. Celles-ci sont également disponibles sur différents systèmes d'exploitation tels qu'Android ou iOS.

FIGURE 3.5 – Partage via les modules des réseaux sociaux sur *lesoir.be*.

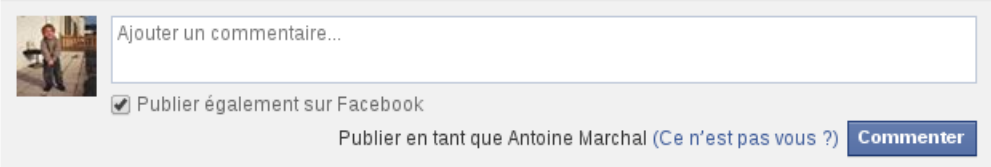

FIGURE 3.6 – Module de commentaires via Facebook sur *lalibre.be*.

FIGURE 3.7 – Module incitant le visiteur à lier son compte Facebook au site *lesoir.be*.

Facebook

Le code que les gestionnaires de sites Web doivent intégrer au sein de leur page afin d'accéder aux fonctionnalités offertes par Facebook [28] est visible à la Figure 3.8. L'intégration de ce code permet aux sites Web d'utiliser entre autres les plugins suivants : le bouton "Like", le bouton de partage, le flux d'actualités d'une page Facebook et le module de commentaires.

Facebook propose également un SDK pour PHP et donne une liste de SDK, plugins et outils développés par d'autres. Les possibilités offertes par Facebook sont nombreuses, elles vont du module d'identification au module de paiement en passant par la publicité. Tout est bien détaillé sur un portail destiné aux développeurs [4].

```
<script>
  window.fbAsyncInit = function() {
        FB.init({
          appId      : '{your-app-id}',
          xfbml      : true,
          version    : 'v2.0'
        });
  };

  (function(d, s, id){
         var js, fjs = d.getElementsByTagName(s)[0];
         if (d.getElementById(id)) {return;}
         js = d.createElement(s); js.id = id;
         js.src = "//connect.facebook.net/en_US/sdk.js";
         fjs.parentNode.insertBefore(js, fjs);
   }(document, 'script', 'facebook-jssdk'));
</script>
```

FIGURE 3.8 – Le SDK de *Facebook* pour JavaScript.

Google+

De son côté, Google propose différents scripts à ajouter en fonction des fonctionnalités que le gestionnaire de site web veut intégrer au sein de sa page (voir Figure 3.9). Google propose même de charger de façon asynchrone son script afin d'obtenir des performances optimales (voir Figure 3.10) [29]. Google+ propose le même genre de plugins [5] que Facebook tels que l'installation d'un module d'identification par compte Google, des boutons de partage, de recommandations, de suivi et autres.

```
<script src="https://apis.google.com/js/plus.js"></script>
<script src="https://apis.google.com/js/client:plus.js"></
   script>
```

FIGURE 3.9 – L'API JavaScript de *Google+*.

4. https://developers.facebook.com/docs/
5. https://developers.google.com/+/

```
1 <script type="text/javascript">
2   (function() {
3     var po = document.createElement('script'); po.type = 'text
        /javascript'; po.async = true;
4     po.src = 'https://apis.google.com/js/plusone.js?onload=
        OnLoadCallback';
5     var s = document.getElementsByTagName('script')[0]; s.
        parentNode.insertBefore(po, s);
6   })();
7 </script>
```

FIGURE 3.10 – L'API JavaScript de *Google+* pour un chargement asynchrone.

LinkedIn

LinkedIn propose également une API pour connecter les sites avec sa plateforme [30]. Dans la Figure 3.11, on peut même voir que le nom de l'utilisateur (s'il est connecté sur la plateforme) est affiché sur le site. Cet aspect semblera convivial à la majorité des utilisateurs mais cela permet surtout de mieux les suivre lors de leur navigation sur les sites intégrant ce module social. Cette fonctionnalité est d'ailleurs également proposée par les autres plateformes de réseaux sociaux.

3.5.3 Outils destinés aux webmasters

Certaines plateformes proposent aux gestionnaires de sites Web de suivre la navigation des visiteurs sur leur site. Elles regardent d'où viennent les visites (moteur de recherche, accès direct, lien d'un autre site,...), leur navigateur (la version, les extensions installées, la langue,...), sur quelles pages les visiteurs se rendent, le temps passé sur chaque page, etc.

Google Analytics (Universal Analytics)

Une des principales plateformes de suivi des utilisateurs est Google Analytics. Cet outil est utilisé par de nombreux gestionnaires de sites afin de connaître les statistiques de fréquentation de leur site [31]. Les webmasters doivent inclure le code JavaScript de la Figure 3.12 afin de pouvoir utiliser cet outil.

Comme on peut le voir sur la Figure 3.13, les statistiques sont très détaillées. Dans la Figure 3.14, la ville des visiteurs était affichée mais il est également possible de voir d'autres données telles que leur FAI ou leur système d'exploitation. Des données spécifiques aux clients mobiles sont également disponibles. Toutes les données sont affichées dans des graphiques clairs et dynamiques et il est même possible d'ex-

```
1  <html>
2  <head>
3  <title>LinkedIn JavaScript API Hello World</title>
4
5  <!-- 1. Include the LinkedIn JavaScript API and define a
       onLoad callback function -->
6  <script type="text/javascript" src="http://platform.linkedin.
       com/in.js">/*
7    api_key: DXjUrY-9
       apre6gnyK080MpTIoIS4f38AleG08Y0kLM0DH2xeNQATIfDMuoisCMO_
8    onLoad: onLinkedInLoad
9    authorize: true
10 */</script>
11
12 <script type="text/javascript">
13   // 2. Runs when the JavaScript framework is loaded
14   function onLinkedInLoad() {
15     IN.Event.on(IN, "auth", onLinkedInAuth);
16   }
17
18   // 2. Runs when the viewer has authenticated
19   function onLinkedInAuth() {
20     IN.API.Profile("me").result(displayProfiles);
21   }
22
23   // 2. Runs when the Profile() API call returns successfully
24   function displayProfiles(profiles) {
25     member = profiles.values[0];
26     document.getElementById("profiles").innerHTML =
27       "<p id=\"" + member.id + "\">Hello " +  member.firstName
           + " " + member.lastName + "</p>";
28   }
29 </script>
30 </head>
31 <body>
32
33 <!-- 3. Displays a button to let the viewer authenticate -->
34 <script type="IN/Login"></script>
35
36 <!-- 4. Placeholder for the greeting -->
37 <div id="profiles"></div>
38
39 </body>
40 </html>
```

FIGURE 3.11 – L'API JavaScript de *LinkedIn*.

porter l'ensemble de ces données dans différents formats (CSV, TSV, Excel, PDF,...) afin de les traiter de manière automatique.

Google a développé une nouvelle version de son outil d'analyse des visiteurs : Universal Analytics. Ce nouvel outil repose sur le même socle que Google Analytics mais il apporte de nouvelles fonctionnalités.

```
<!-- Google Analytics -->
<script>
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r
    ]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.
    createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.
    insertBefore(a,m)
})(window,document,'script','//www.google-analytics.com/
    analytics.js','ga');

ga('create', 'UA-XXXX-Y', 'auto');
ga('send', 'pageview');

</script>
<!-- End Google Analytics -->
```

FIGURE 3.12 – L'API JavaScript de *Google Analytics*.

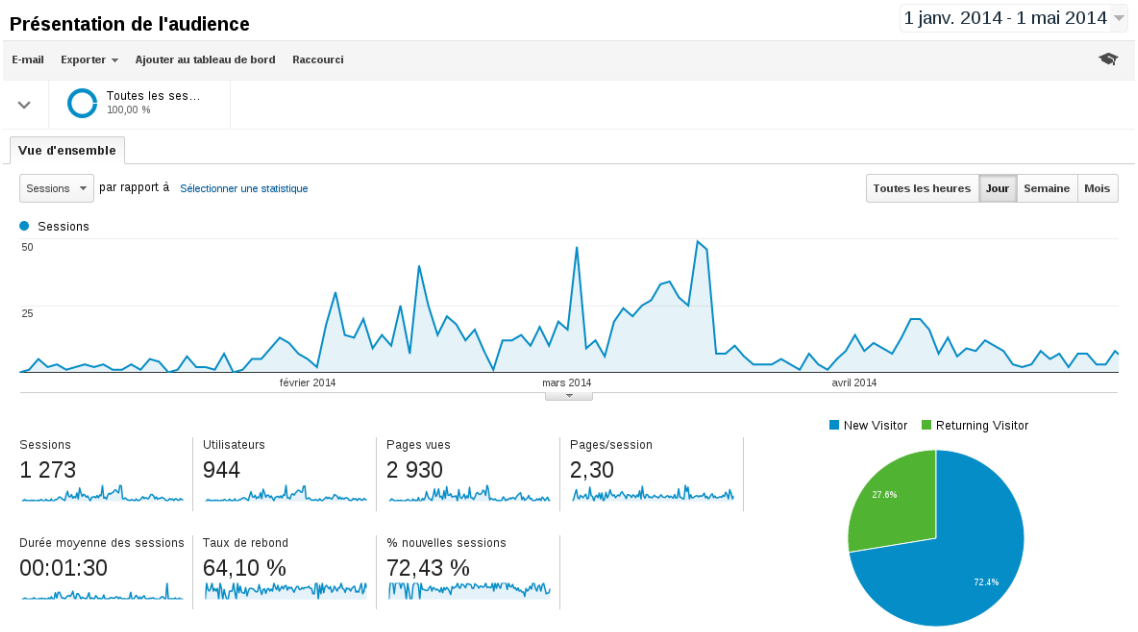

FIGURE 3.13 – Copie d'écran de *Google Analytics* qui détaille les visites.

FIGURE 3.14 – Copie d'écran de *Google Analytics* qui détaille l'origine des visiteurs.

Pour ce nouvel outil, Google propose 3 types de code de tracking en fonction des plateformes visées : la librairie JavaScript *analytics.js* pour les sites Web, les SDK Google Analytics pour les applications mobiles et le *protocole de mesure ("Measurement Protocol")* pour les autres appareils tels que consoles de jeux.

Alors que Google Analytics identifiait un utilisateur différent en fonction de chaque appareil connecté, Universal Analytics reconnaît un même utilisateur qui utilise plusieurs moyens de se connecter à Internet. Afin d'y parvenir, Google utilise un identifiant d'utilisateur unique afin d'associer les données reçues d'appareils et de sessions différents.

Afin d'utiliser cette fonctionnalité, les gestionnaires de sites Web doivent être en mesure de générer un identifiant unique pour chaque utilisateur et l'associer aux données envoyées à Google. Généralement, cet identifiant unique est généré grâce à l'authentification des visiteurs sur un site. Ainsi, lorsqu'un utilisateur se sert de sa tablette et de son ordinateur pour se rendre sur un site et s'il s'y connecte, Google sera en mesure d'identifier ces visites comme provenant d'un seul et même utilisateur et non plus de deux utilisateurs différents.

En fonction de la version de Google Analytics utilisée, différents cookies sont créés et utilisés afin de suivre les visites des utilisateurs [16].

La nouvelle librairie JavaScript de Google, *analytics.js* crée un cookie d'origine contenant un identifiant anonyme afin de distinguer les utilisateurs. Par défaut, la librairie crée un cookie sur le domaine de plus haut niveau et règle le chemin du cookie sur le niveau racine.

Le cookie créé est :

- **__ga** : cookie avec une date d'expiration de 2 ans destiné à distinguer les utilisateurs.

L'ancienne libraire JavaScript de Google, *ga.js*, crée 5 cookies d'origine qui permettent de :

— déterminer quel domaine est mesuré
— distinguer les utilisateurs uniques
— se souvenir du nombre et des heures des visites précédentes
— se souvenir de l'information sur la source de trafic
— déterminer le début et la fin de session
— se souvenir des valeurs des variables personnalisées au niveau de l'utilisateur

Par défaut, la librairie crée un cookie sur le domaine spécifié dans la propriété du navigateur *document.host* et règle le chemin du cookie sur le niveau racine.

Les cookies créés sont :

- **___utma** : cookie avec une date d'expiration de 2 ans destiné à distinguer les utilisateurs et les sessions.
- **___utmb** : cookie avec une date d'expiration de 30 minutes destiné à distinguer les nouvelles sessions et visites.
- **___utmc** : cookie avec une date d'expiration de fin de session qui n'est plus utilisé dans *ga.js* mais qui permet l'interopérabilité avec *urchin.js* (une librairie encore plus ancienne de Google Analytics, antérieure à *ga.js*).
- **___utmz** : cookie avec une date d'expiration de 6 mois destiné à enregistrer les sources de trafic ou de campagne qui retrace comment un utilisateur atteint le site.
- **___utmv** : cookie avec une date d'expiration de 2 ans utilisé pour sauvegarder les données des variables personnalisées au niveau de l'utilisateur.

Avec le nombre élevé de sites utilisant les services de Google, ce dernier a les moyens de dresser une carte d'Internet. Il est même en mesure de connaître la plupart des sites visités par l'utilisateur (à la condition qu'ils intègrent un tracker de Google). Cela va sans dire que Google détient une très grande quantité d'informations sur les utilisateurs du Web actuel quand on y ajoute les messages privés (Gmail), la localisation (Android) et les nombreux autres services du groupe.

3.6 Flash

Adobe Flash Player [32] est un logiciel qui permet de lire du contenu multimédia. Il est notamment utilisé pour la conception de jeux sur Internet ou pour le streaming audio/vidéo (il était auparavant utilisé par YouTube qui a désormais décidé de se tourner vers la technologie HTML 5 [33]).

Flash peut être utilisé dans le tracking des utilisateurs car il utilise son propre système de cookies : les LSO (Local Shared Objects) [34]. Ces derniers sont même considérés plus invasifs que les cookies HTTP car ils possèdent ces caractéristiques qui les rendent spéciaux [35] :

- ils permettent de stocker plus d'informations (100KB au lieu de 4KB pour les cookies HTTP)
- ils n'ont pas de date d'expiration par défaut
- ils sont stockés dans un emplacement différent, ce qui implique que la plupart des utilisateurs ne savent pas les effacer facilement et qu'ils sont accessibles via différents navigateurs
- ils ne sont pas contrôlés par le navigateur et celui-ci ne peut donc pas les effacer
- ils permettent de stocker des types de données simples (String, Array et Date)

Ces caractéristiques font que les cookies Flash (ce sont des fichiers avec l'extension *.sol*) sont plus persistants et méconnus des utilisateurs. Pour supprimer les cookies Flash sans connaître leur emplacement, il faut se rendre sur une page dédiée [6] du site d'Adobe [7].

Le répertoire utilisé pour les stocker sur Linux est : */home/<user>/.macromedia/Flash_Player*. Celui-ci abrite deux répertoires :

1. macromedia.com : il contient les paramètres de chaque site utilisant Flash.
2. #SharedObjects : il contient les LSO (les cookies Flash).

Avec le contenu du répertoire *macromedia.com*, on peut constater un autre risque de fuites concernant la vie privée : en analysant les traces laissées dans ce répertoire, il est possible de reconstituer une partie de l'historique de navigation de l'utilisateur. En effet, pour chaque domaine, un dossier est créé portant son nom et contenant un fichier de préférences *settings.sol*. Simplement en regardant le nom des dossiers, il est possible de découvrir certains sites visités (voir Figure 3.15).

6. `http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager07.html`
7. Adobe est le développeur de Flash Player.

Notez que le navigateur Google Chrome n'utilise pas ce répertoire car Flash est intégré au navigateur [36]. Il est donc possible de supprimer les cookies Flash dans les paramètres du navigateur ou en se rendant sur la page dédiée du site Adobe.

```
1 tibesti:~/.macromedia/Flash_Player/macromedia.com/support/
      flashplayer/sys antmarchal$ ls
2 #a1.itc.cn              #cdn.flashtalking.com  #effectivemeasure.net
3 #acjs.aliyun.com        #cdn.gotraffic.net     #graphics.wsj.com
4 #aka-cdn-ns.adtech.de   #cdnl3.neulion.com     #heias.com
5 #a.tbcdn.cn             #cfiles.5min.com       #i0.poll.fm
6 #cdn1.img22.ria.ru      #chaturbate.com        ...
7 #cdnbakmi.kaltura.com   #content.adriver.ru
```

FIGURE 3.15 – Affichage des dossiers dans *macromedia.com*.

Dans une expérience réalisée en 2009 [34], il a été montré que dans le TOP 100 des sites (classement Quantcast - juillet 2009), 54 sites ont utilisé des cookies Flash. Ces 54 sites ont généré 157 fichiers LSO pour arriver à un total de 281 cookies individuels. 98% de ces sites ont également créé des cookies HTTP (seuls *wikipedia.org* et *wikimedia.org* ne l'ont pas fait) pour un total de 3602 cookies HTTP.

Ensuite, les chercheurs ont analysé les noms de variables les plus fréquemment utilisés dans ces cookies. Il en est ressorti que "volume" était en première position, ce qui fait penser à une préférence de volume pour le streaming (ce qui est logique car Flash reste avant tout un lecteur multimédia). Juste derrière et dans la majorité des cas, les noms de variables reflétaient l'idée d'identifiant (*userid*, *user*, *id*,...). Il est donc clair que les cookies Flash sont utilisés pour tracer les utilisateurs.

Le contenu des cookies Flash a été comparé avec le contenu des cookies HTTP. Sur les 100 sites, 31 montraient au moins un lien entre les cookies Flash et HTTP. Sur la totalité des 31 sites, 41 correspondances ont été trouvées.

Les chercheurs ont également montré que les cookies Flash étaient utilisés pour faire réapparaître les cookies HTTP lorsqu'ils étaient supprimés. Ainsi, certains sites recréaient les cookies HTTP grâce aux cookies Flash. Donc même si l'utilisateur effaçait les cookies HTTP via les préférences de son navigateur, les cookies de certains sites avec leur valeur étaient de nouveau envoyés lors des visites sur ces sites. Il faut également noter que certains cookies HTTP étaient recréés par des cookies Flash tiers.

En 2011, l'expérience a été menée à nouveau [25]. Cette fois, les chercheurs se sont penchés également sur le stockage local HTML5 (voir la sous-section 3.3.2). Les résultats sont les suivants : 5675 cookies HTTP ont été détectés (au lieu de 3602 en 2009). 20 sites ont placé 100 cookies ou plus et 7 sites en ont placé plus de 150. La plupart des noms de variables de ces cookies font penser à un identifiant unique de tracking. La plupart des cookies (4915) sont placés par un domaine tiers.

Concernant les cookies Flash, ils sont en baisse car 100 ont été détectés (au lieu de 281 en 2009) sur 37 sites (au lieu de 54 en 2009). Deux sites (hulu.com et foxnews.com) utilisaient des valeurs communes dans leurs cookies Flash et HTTP, la valeur était également commune dans le stockage local HTML5.

Il faut noter qu'Adobe a condamné l'utilisation des LSO à des fins de tracking. L'entreprise incite les développeurs à utiliser la technologie Flash de manière responsable et de ne pas utiliser de technique qui permette d'utiliser le stockage local afin de recréer des cookies effacés par l'utilisateur sans son consentement [8]. Ils détaillent également quelques nouveautés qui ont été ajoutées aux versions ultérieures de Flash (notamment le support de la navigation privée).

Avant la version 10.1 de Flash Player [37], celui-ci ne prenait pas en compte les effets des sessions de navigation privée des navigateurs. Cela signifie que lorsqu'un utilisateur activait une session de navigation privée avec son navigateur, les cookies, fichiers en cache et l'historique de sa session étaient supprimés par le navigateur mais ce n'était pas le cas pour les cookies Flash. Depuis la version 10.1 de Flash Player, celui-ci garde les fichiers LSO de la navigation privée en mémoire sans les enregistrer sur le disque et n'accède pas non plus aux LSO existants. A la fin de la session de navigation privée, Flash Player efface les données en conformité avec les préférences du navigateur.

3.7 Empreintes des navigateurs

Jusqu'à présent, nous avons vu des techniques de tracking dites avec état (stateful tracking). A l'opposé, il existe des techniques dites sans état (stateless tracking) utilisant des *fingerprinters* [9] [38]. Ces derniers récupèrent une série d'informations qu'ils agrègent afin d'associer un identifiant (presqu') unique à l'utilisateur. Le principal avantage de cette méthode est que les informations dépendent peu de l'utilisa-

8. La déclaration complète d'Adobe est disponible à l'URL suivante : `http://www.ftc.gov/policy/public-comments/comment-544506-00085`

9. En français, on pourrait traduire ce mot par "calculateur d'empreinte".

teur. En effet, avec des techniques utilisant des cookies, le tracking peut être rendu complexe par la suppression des cookies par l'utilisateur. A l'inverse, avec les fingerprinters, les informations récupérées sont relativement stables grâce à leur nature (numéro de version du navigateur, système d'exploitation, résolution d'écran, etc).

Le *fingerprinter* regroupe les informations qui lui sont nécessaires via différentes sources [39]. La première source est constituée des plugins populaires tels que Flash. Par exemple, lorsqu'on demande au navigateur la plateforme d'exécution sur une version 64 bits de Linux, il répond "Linux x86_64" alors que si on demande la même information à Flash, il répond en indiquant la version complète du noyau Linux. Il est également possible de déterminer si l'utilisateur utilise un second écran. En effet, avec l'implémentation Linux du plugin Flash, lorsque la résolution est demandée, Flash renvoie comme résolution d'écran la somme des écrans individuels alors que le navigateur renvoie la résolution de l'écran sur lequel la fenêtre est affichée.

Les outils de fingerprinting s'adaptent également aux navigateurs des utilisateurs. Il a été montré qu'ils utilisent des propriétés spécifiques à certains navigateurs afin de récupérer certaines données. C'est notamment le cas avec Internet Explorer et les propriétés *navigator.securityPolicy* et *navigator.systemLanguage* [39].

Une autre source importante d'informations vient des polices de caractères utilisées. Cette information n'est pas directement disponible via les navigateurs et les *fingerprinters* n'hésitent pas à utiliser des plugins comme Flash pour récupérer cette liste. De plus, certains *fingerprinters* incluent un "plan B" lorsque Flash n'est pas disponible. Ils mesurent la taille d'un phrase prédéfinie avec la police utilisée par défaut si la police demandée n'est pas disponible. Ensuite, ils appliquent une police différente et mesurent la taille de la même phrase. Si la taille est identique, cela veut dire que la police n'est pas disponible chez l'utilisateur. Ils sont ainsi en mesure de déterminer une liste de polices installées chez l'utilisateur (avec l'inconvénient que les polices les plus rares ne sont pas détectées). Il faut noter que la présence de certaines polices rares permettent d'identifier un utilisateur avec une précision beaucoup plus importante [40].

Flash est également utilisé pour détecter les proxys HTTP. C'est d'ailleurs la raison pour laquelle il est désactivé dans des applications sensibles à l'anonymat.

Une dernière source utilisée dans Internet Explorer provient des DLL chargées dans le navigateur. Certaines accèdent à des valeurs du registre Windows afin de transmettre des informations telles que les identifiants des disques durs, des paramètres TCP/IP, le nom de l'ordinateur, l'identifiant de produit d'Internet Explorer, la date d'installation de Windows, l'ID produit Windows Digital et les pilotes [39].

Au sein du tracking sans état, on peut distinguer deux types d'utilisation du *fingerprinting* : le *fingerprinting actif* qui acquiert les informations de manière active grâce à l'utilisation de scripts ou de plugins et le *fingerprinting passif* qui acquiert les informations de manière passive grâce au trafic réseau (les requêtes et réponses HTTP). Le *fingerprinting passif* est particulièrement problématique car il n'est pas détectable par le client. Un fingerprinter n'utilisant que du fingerprinting passif sera moins performant mais sera néanmoins en mesure de tracer certains types d'utilisateurs qui visitent le Web avec un navigateur qui est rarement modifié (pas d'installation d'extensions) et peu mis à jour.

Un aspect intéressant à analyser est la stabilité des empreintes des navigateurs. En effet, l'empreinte d'un navigateur peut être modifiée par différentes actions : mettre à jour le navigateur, mettre à jour un plugin, désactiver les cookies, installer une nouvelle police de caractères (ou une application qui contient des polices) ou connecter un second moniteur (ce qui change la résolution d'écran). Il semblerait que les empreintes calculées ne soient pas particulièrement stables mais le fait que le navigateur dévoile tant d'informations rend cette technique de tracking très intéressante. [41]

3.8 Conclusion

Les moyens existants et mis en place dans le but de tracker les utilisateurs sont nombreux et divers. Certains restent relativement basiques alors que d'autres utilisent des techniques plus complexes. Certains sont conçus principalement dans le but d'effectuer un tracking sur un site particulier (les outils d'analyses de visites) alors que d'autres sont destinés à tracer les visiteurs sur de multiples sites (les régies publicitaires) afin de créer un profil des visiteurs.

Il semble que les possibilités et moyens de tracker les utilisateurs naviguant sur Internet évoluent rapidement. De nouvelles techniques apparaissent alors que d'autres sont peu à peu oubliées.

Chapitre 4

Analyse des principaux sites web

4.1 Objectif de l'analyse

Le but de l'analyse est de parcourir les sites les plus visités au monde d'après le classement Alexa [1]. Ensuite, déterminer s'ils contiennent des trackers et, le cas échéant, leur nature. Deux types d'expérience peuvent être menés :

1. Le premier consiste en une analyse à long terme.
2. Le second consiste en une analyse ponctuelle.

Le premier type d'expérience a pour but de déterminer si les sites modifient leurs trackers au fil du temps. Le second type permet d'avoir une image globale du niveau de traçage opéré par les principaux sites à un instant donné. Il donne également la possibilité de tester l'efficacité des extensions de navigateur qui affirment protéger la vie privée (voir section 6.4).

4.2 Description de l'outil implémenté

L'idée initiale était d'utiliser le framework *fpdetective* [43]. C'est un framework conçu pour la détection et l'analyse d'outils (*fingerprinters*, voir section 3.7) qui créent des empreintes de navigateurs. Ce framework semblait prometteur mais son efficacité s'est révélée insuffisante et il ne fonctionnait pas correctement. En effet, aucun *fingerprinter* n'était détecté alors que certains étaient manifestement présents sur plusieurs sites web visités.

Étant donné que *fpdetective* ne pouvait fournir les résultats demandés, la création d'un outil dédié à cette tâche était nécessaire. De plus, cela permettait davantage

1. Alexa [42] est une entreprise qui appartient à Amazon et qui est principalement connue pour les statistiques qu'elle fournit sur le trafic Web dans le monde entier.

de souplesse au niveau des décisions d'implémentation et du paramétrage de la recherche de trackers.

L'outil a été implémenté en Java. Il se compose de deux éléments : le *crawler* qui visite les sites web et exporte leur contenu au format HTTP Archive (HAR) et le *parser* qui traite les fichiers HAR en déterminant si les sites correspondants renferment des trackers. L'outil a été développé sous Eclipse et est destiné à être utilisé sur Linux. Un exécutable JAR est également disponible. La liste des options disponibles au lancement de l'outil est détaillé dans l'Annexe A.

4.2.1 Crawler

Choix d'implémentation

Afin d'automatiser le navigateur et de parcourir les sites sans intervention humaine, *Selenium*[2] est utilisé dans le *crawler* ou plus précisément, le pilote *Firefox-Driver* de *Selenium*. Il permet de lancer une instance de Firefox et d'interagir avec celle-ci.

La première implémentation traitait directement le code source de la page. Il était possible d'aller sur le moteur de recherche Google, de sélectionner le champ de recherche, de taper une requête et de récupérer le résultat. Cependant, le processus était assez lent et incomplet car on ne pouvait pas récupérer les requêtes et les réponses HTTP. On pouvait juste collecter les éléments associés à des balises (par exemple, récupérer toutes les images via les balises de type *<img>*, les scripts avec les balises *<script>*,...) mais tout n'était pas récupérable car certains éléments étaient chargés sans être identifiés.

Afin de pallier à ce problème, la deuxième implémentation utilisait un proxy[3] qui exportait le contenu du site au format HTTP Archive[4]. L'ensemble des requêtes et réponses HTTP était récupéré et enregistré dans un fichier. Le parcours du code source de la page (qui donnait des résultats finalement incomplets) a été supprimé afin de rendre le programme plus rapide. Cette fois, le problème était que le proxy n'exécutait pas le JavaScript. Il en résultait que de nombreux sites ne pouvaient

2. *Selenium* [44] est une suite d'outils qui permet d'automatiser les navigateurs. On peut par exemple créer des scripts qui vont faire visiter un navigateur sur différents sites et effectuer plusieurs actions.

3. Un proxy est un serveur qui sert d'intermédiaire afin d'accéder à un réseau et récupérer des ressources d'un autre serveur.

4. HTTP Archive est un format ouvert qui permet d'exporter des échanges de données collectés par un outil de monitoring HTTP [45].

être chargés correctement (cela pouvait varier de 15 à 30% d'échecs) et il était alors impossible de les traiter.

La troisième solution, qui est la dernière implémentée, consiste à utiliser le pilote *FirefoxDriver* pour charger automatiquement les sites, sans parcours du code source ni utilisation de proxy. Le contenu des sites est exporté au format HTTP Archive à l'aide de deux extensions qui ont été installées dans Firefox.

— La première, Firebug [46], est une extension utilisée principalement par les développeurs de sites web afin de les débugger. Elle permet de monitorer les éléments d'une page (CSS, HTML, JavaScript) en direct.

— La seconde, NetExport [47], est une extension de Firebug qui permet d'exporter toutes les données d'une page au format HTTP Archive de façon automatique moyennant un paramétrage adéquat.

Le taux de réussite de l'export des données des sites s'est grandement amélioré. Néanmoins, certains sites ne sont toujours pas chargés correctement. Une hypothèse est que certains sites sont hébergés sur des serveurs lointains et leurs éléments prennent alors plus de temps à charger. Ainsi, lorsque ces sites contiennent une grande quantité d'éléments, le pilote renvoie une exception de type *timeout* et le chargement du site est finalement en échec.

Afin de vérifier cette hypothèse, une série de tests a été mise en place. Le test consistait à faire varier la valeur du timeout et de vérifier si cela influençait le nombre de sites en échec avec le *crawler*. Celui-ci a donc été lancé sur le TOP 500 avec des valeurs de timeout différentes. Le graphique est visible à la Figure 4.1 et il montre que cette hypothèse est vérifiée.

Une amélioration a été apportée à l'outil afin de compter le nombre de cookies Flash créés lors de chaque visite de site. Cette fonctionnalité a été implémentée au sein du *crawler* étant donné que ce n'est pas possible d'accéder à cette information avec l'aide des fichiers HTTP Archive analysés par le *parser*. Cela signifie qu'une partie des résultats est déjà disponible lorsque le *crawler* a fini de visiter les sites.

A son lancement, il supprime l'ensemble des cookies Flash (les fichiers .sol) dans le dossier *.macromedia/Flash_Player/#SharedObjects/XXXXXXXX/*. Il supprime également les cookies Firefox en se connectant à la base de données SQLite dans *.mozilla/firefox/<profil>/cookies.sqlite* et en vidant la table *moz_cookies*. Cela permet de s'assurer que la visite des sites ne sera pas influencée par la présence de cookies antérieurs.

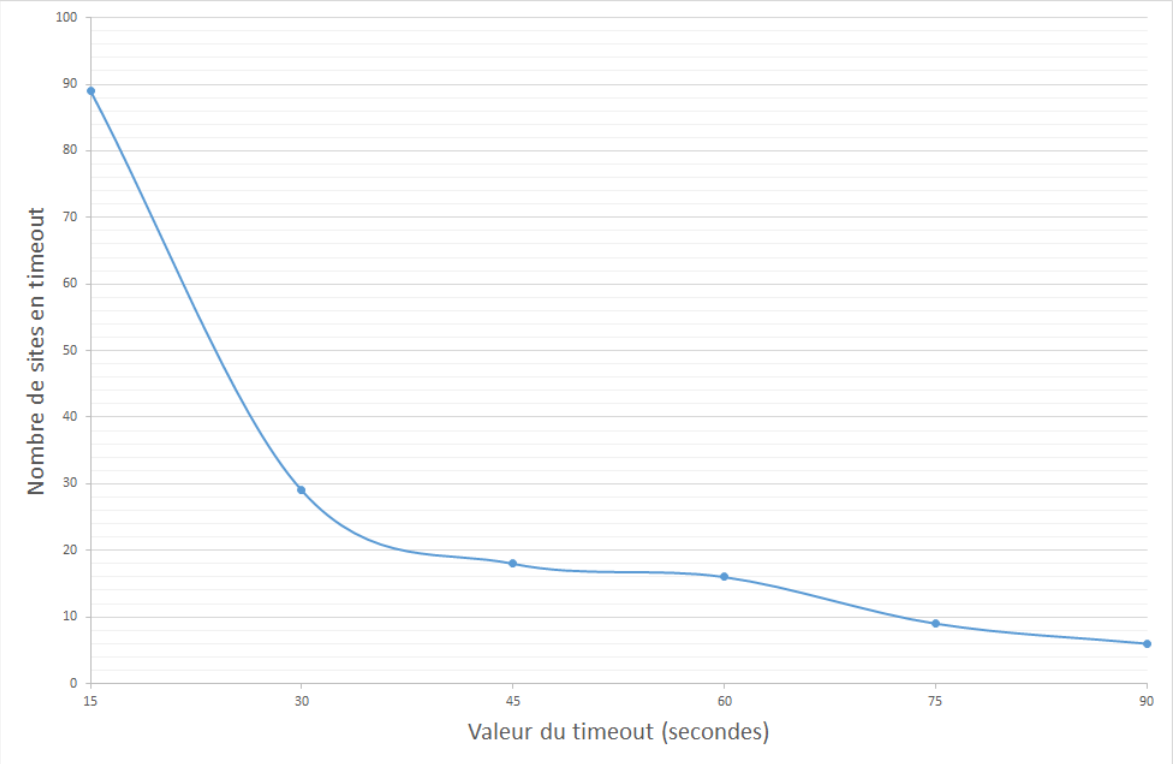

FIGURE 4.1 – Nombre de sites en échec en fonction du timeout sur le TOP 500.

Changements opérés sur les extensions

Deux modifications ont été nécessaires sur l'extension *NetExport* de *Firebug*. Les changements dans leur code source sont détaillés dans l'Annexe B.

La première concerne la génération des fichiers HTTP Archive. Par défaut, l'extension ajoute un champ qui n'est pas standard par rapport à la spécification du format HAR [45]. Il en résulte que les fichiers peuvent alors être considérés comme corrompus alors que la présence ou non de ce champ n'influence en rien les résultats qui nous concernent. La génération de ce champ a donc été supprimée dans le fichier */chrome/content/netexport/harBuilder.js* de *NetExport*.

La seconde modification concerne le nom des fichiers générés. Par défaut, l'extension ajoute la date à l'URL du site dans les noms de fichier. Cela pose des difficultés supplémentaires au *parser* lors de la récupération et du traitement de ces fichiers (voir sous-section 4.2.2). L'extension a donc été modifiée afin de n'utiliser que l'URL du site dans les noms du fichier. Cette modification a été effectuée dans le fichier */chrome/content/netexport/automation.js*. Sur Linux, lorsqu'un fichier de même nom existe déjà, un tiret suivi du numéro de copie est ajouté au nom de fichier avant son extension.

Fonctionnement

Le *crawler* fonctionne de la manière suivante. D'abord, il vérifie que le répertoire donné en argument est accessible en écriture et il crée deux dossiers nommés "logs" et "results" qui sont destinés à contenir respectivement les logs et les résultats du *crawler*. Ensuite, le fichier de log (log_crawler.txt) est créé, le dossier des cookies Flash est détecté (s'il y en a plusieurs, l'utilisateur est invité à indiquer le bon dossier) et la base de données SQLite des cookies Firefox est vidée (le dossier des paramètres de l'utilisateur est automatiquement trouvé grâce au profil Firefox spécifié en argument au lancement du programme). La liste des sites à visiter est ainsi chargée depuis le fichier spécifié et en fonction de l'intervalle donné par l'utilisateur. Puis, le profil Firefox choisi est chargé et différents paramètres de configuration des extensions sont également réglés avant que le pilote ne soit initialisé et qu'il lance Firefox. Après le lancement du pilote, un délai de 5 secondes laisse le temps à Firefox de se lancer et de charger ses extensions.

Après cette phase de préparation, le parcours des sites commence. Le pilote dirige Firefox sur l'URL du premier site de la liste. Lorsque le chargement de la page est terminé, un délai de 8 secondes permet aux extensions d'enregistrer le fichier HTTP Archive. Le nombre de cookies Flash est enregistré pour chaque site.

Si la page s'est chargée avant la limite de timeout définie par l'utilisateur (par défaut, 30 secondes), l'enregistrement est considéré comme valide et le pilote charge le site suivant dans la liste.

Si un site fait un timeout et que l'utilisateur a précisé un nombre de tentatives supérieur à 1, le pilote dirige Firefox sur la page "about :blank" (cela permet d'éviter des problèmes d'écriture de fichiers avec les extensions) puis le redirige sur le site pour un nouvel essai. Ce processus est répété à chaque timeout jusqu'à atteindre le nombre maximal de tentatives. Si le site fait encore un timeout lors de la dernière tentative, il est alors considéré en échec et son URL est inscrite dans une liste reprenant tous les sites en échec. A la deuxième tentative, l'URL du site a déjà été enregistrée dans une autre liste reprenant tous les sites ayant fait un timeout, cela permet de garder une trace des sites dont la visite a rencontré un problème mais qui ont finalement pu être chargés.

Il arrive également qu'un site ne puisse tout simplement pas être chargé (à cause d'une erreur dans son code source par exemple). Son URL est alors directement enregistrée dans la liste reprenant les sites en échec.

Lorsque le parcours de l'ensemble des sites est terminé, le programme efface les fichiers inutiles générés lors de la visite des pages "about :blank" et détaille l'ensemble des sites ayant subi au moins un timeout et les sites en échec. La liste des sites avec leur nombre de cookies Flash est enregistrée en ordre décroissant dans le fichier "stats_flash-cookies.csv" du dossier "logs". Pour terminer, le pilote est arrêté (ce qui entraîne la fermeture de Firefox) et le fichier de log est fermé.

Il a été nécessaire d'implémenter une fonctionnalité qui redémarre Firefox à intervalles réguliers. En effet, certains sites ouvrent des pop-ups et si le navigateur n'est pas fermé, ces derniers s'accumulent et occupent une part plus importante de ressources. Le redémarrage du navigateur permet de fermer toutes ces fenêtres ouvertes de façon intempestive afin de garantir une meilleure stabilité du *crawler*.

4.2.2 Parser

Choix d'implémentation

Lors de la visite des sites par le *crawler*, il arrive que celui-ci doive refaire de nouvelles visites afin d'exporter correctement le fichier HTTP Archive d'un site. Il en résulte que plusieurs fichiers pour le même site soient disponibles. Par exemple, on peut très bien voir des fichiers tels que *monsite.com.har*, *monsite.com-1.har*, etc. Lors de l'analyse de ces fichiers par le *parser*, celui-ci considère seulement la dernière version du fichier comme étant valide. Il ignore donc les autres fichiers concernant le même site (si jamais il y en a plusieurs) car ils sont généralement corrompus ou incomplets.

Lors de l'analyse d'un fichier, l'URL du site est extraite (c'est le nom du fichier) et l'autorité en charge pour son domaine est automatiquement récupérée via une requête DNS de type SOA. Cela permettra de déterminer si une URL présente dans le site appartient au même domaine ou si elle provient d'un domaine différent.

Si l'autorité du domaine du site analysé ne peut être récupérée, l'analyse du site est considérée comme étant en échec et le nom du fichier en cours d'analyse est ajouté dans une liste reprenant tous les fichiers en échec.

Si l'autorité du domaine d'une URL analysée au sein d'un fichier ne peut être récupérée, l'analyse de cette URL est passée mais l'analyse des URL suivantes du fichier continue. Un message est imprimé dans le log pour préciser que la récupération de l'autorité du domaine de l'URL n'a pas pu être récupérée.

Lorsqu'une URL est constituée d'une adresse IP, l'outil fait une requête DNS afin

de connaître le nom associé à cette IP. S'il n'y parvient pas, l'analyse de l'URL est passée mais le processus d'analyse continue comme c'est le cas lorsqu'une autorité ne peut être récupérée pour une URL. Ceci est dû au fait que la libraire utilisée (*dnsjava*) pour effectuer les requêtes DNS n'est pas en mesure d'effectuer des requêtes SOA pour des adresses IP.

Le fonctionnement de la récupération de l'autorité d'une URL est le suivant : tout d'abord, la requête est effectuée sur l'hôte de l'URL. Si aucune réponse n'est retournée, la requête est effectuée sur le domaine parent et ainsi de suite, jusqu'à trouver une réponse ou arriver au domaine le plus haut (voir Figure 4.2).

Un système de mise en cache a été mis en place afin de limiter le nombre de requêtes effectuées et d'améliorer la rapidité d'accès à l'information.

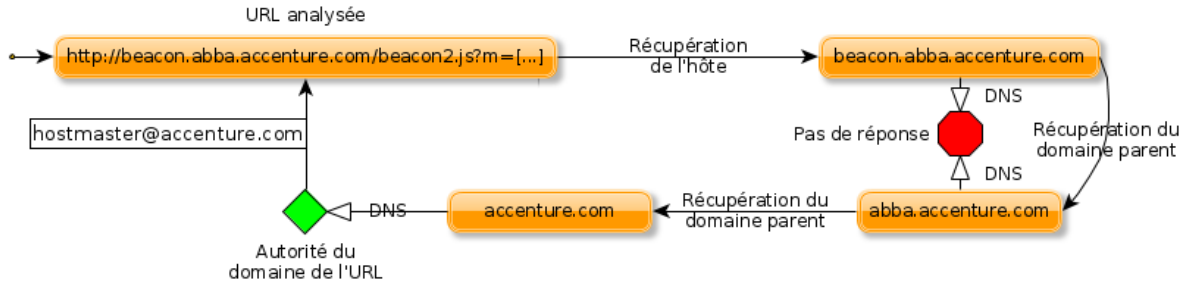

FIGURE 4.2 – Schéma de récupération de l'autorité du domaine.

Limitation des requêtes DNS de type SOA

Le recours aux informations SOA permet généralement de déterminer si deux sites appartiennent au même domaine. C'est ainsi que lors de la visite de *youtube.com*, les images provenant de *ytimg.com* ne sont pas détectées comme provenant d'un site externe. En effet, si on effectue deux requêtes DNS de type SOA pour *youtube.com* et *ytimg.com*, la même autorité de domaine est renvoyée : *dns-admin@google.com*. Dans ce cas, toute URL présente dans le site *youtube.com* provenant de *ytimg.com* ne sera pas considérée comme extérieure au site.

Cependant, il peut y avoir des faux positifs résultant des réponses reçues. Lors des analyses, un phénomène a été remarqué : à certains moments, la requête DNS peut renvoyer une réponse différente. Ceci a été observé pour le domaine *p8.qhimg.com*. Dans ce cas précis, la requête DNS précise que l'autorité du site est en fait celle de l'hébergeur de contenu *cloudfront.net* (qui est lui-même hébergé sur les serveurs

d'*Amazon*) et non plus celle du site visité. Afin d'éviter ce problème, il serait possible d'utiliser une liste des différentes autorités mise à jour en dehors du programme et vérifiée de façon indépendante et périodique. Cela n'a pas été jugé nécessaire lors de l'implémentation car le problème survient assez rarement (lors du lancement du *parser* sur l'analyse de 1000 sites, cette situation s'est produite à seulement 2 reprises).

Critères qualifiant un tracker

Le *parser* parcourt l'ensemble des requêtes et réponses HTTP effectuées lors du chargement du site. Afin de déterminer si une ressource chargée est un tracker, l'outil se base sur la base de données Ghostery et sur plusieurs critères qui sont détaillés plus tard.

Ghostery [5] est une extension disponible pour différents navigateurs. Son fonctionnement repose sur une base de données de trackers alimentée par les retours des utilisateurs de l'extension. Afin de pouvoir utiliser la base de données Ghostery, un contact a été pris avec Evidon (les propriétaires de l'extension) qui a donné son autorisation d'utiliser la base de données au sein de l'outil.

Si l'utilisateur donne en argument un fichier contenant la base de données Ghostery, le *parser* cherche en premier lieu une correspondance entre l'URI d'une requête et un tracker connu par Ghostery à l'aide d'expressions régulières. Si l'URI est connue pour être un tracker, elle est enregistrée dans une liste spécifique aux trackers détectés par Ghostery et le type MIME [6] de l'élément est enregistré.

Si aucune correspondance n'a pu être trouvée ou si l'utilisateur n'a pas spécifié de fichier contenant la base de données Ghostery, l'autorité du domaine hébergeant la ressource demandée est récupérée et comparée à celle du site visité. Si les autorités sont différentes, les requêtes et réponses HTTP vont être inspectées plus en détail afin de déterminer si elles constituent des appels à des trackers.
Des données sont enregistrées dans les cas suivants :

- La ressource chargée est du JavaScript : inclure un code JavaScript dans sa page lui permet de passer outre le principe de même origine, il y a donc un danger potentiel (section 3.5).
- La ressource chargée est du JavaScript et l'URL contient des paramètres (chaîne de requête) : le script importé est vraisemblablement un tracker.

5. `https://www.ghostery.com/`
6. Le type MIME (*mimetype* en anglais) est un identifiant de format de données.

— La ressource chargée est du Flash : Flash permet de créer des cookies persistants (section 3.6) et donne accès à des informations normalement indisponibles au navigateur (section 3.7).
— Les images chargées ont une largeur et hauteur de 1 pixel : il s'agit de pixels espions (section 3.4).
— Un cookie est créé via la réponse HTTP : une réponse provenant d'un domaine tiers ne devrait pas créer de cookie (section 3.2).
— L'URL des ressources chargées contient des paramètres (chaîne de requête) : il y a danger car une requête effectuée vers un domaine tiers permet de transmettre des informations (section 3.1).

Ces règles sont appliquées dans l'ordre avec lequel elles sont énumérées ci-dessus. D'abord, le type de la ressource est analysé, ensuite c'est la création d'un cookie tiers et pour terminer, la présence d'une requête dans l'URI. L'ordre de ces critères a une influence : la majorité des pixels de tracking créent un cookie. Or, si on place le critère de création de cookie avant l'analyse du type de la ressource, le tracker sera classifié comme réponse tierce qui crée un cookie et non comme un pixel de traçage. L'élément serait dans les deux cas identifié comme un tracker mais il ne se trouverait pas dans la même catégorie.

Toutes ces données sont enregistrées dans des fichiers différents spécifiques à chaque site afin de permettre une analyse précise. De plus, un fichier reprend la liste des sites analysés ainsi que le nombre d'éléments enregistrés pour chaque règle. Il est également possible de faire des statistiques sur les critères de façon indépendante (voir le format des fichiers dans l'Annexe C).

Il n'est pas possible de déterminer si les images provenant de régies publicitaires constituent des trackers sans utiliser de liste noire prédéfinie. En effet, les pixels espions sont facilement détectables étant donné leur taille spécifique mais il n'existe pas de points de différenciation entre les images affichées qui constituent un tracker et les images qui ne sont pas liées au tracking. Cependant, il est très probable que les images affichées depuis les serveurs des régies publicitaires jouent le rôle de tracker.

Normalement, de tels trackers devraient être détectés par Ghostery. Les régies publicitaires jouant au chat et à la souris avec les listes noires de trackers, ces derniers ne sont parfois pas détectés car leur signature n'a pas encore été intégrée aux bases de données utilisées pour leur détection.

Fonctionnement

Le but du *parser* est de traiter les fichiers HTTP Archive générés par le *crawler* et de déterminer le nombre de trackers pour chaque site. La première chose qu'il fait à son lancement est de vérifier que le dossier donné en argument est accessible en écriture. Il regarde ensuite si des dossiers "logs" ou "results" sont déjà présents et si ce n'est pas le cas, il les crée. S'ils sont déjà présents, le *parser* avertit l'utilisateur qu'il va les utiliser afin d'y écrire des données. Concernant le dossier "results", il demande à l'utilisateur s'il souhaite continuer car des fichiers provenant de résultats calculés antérieurement pourraient être écrasés. Puis, le *parser* ouvre un fichier de log ("log_parser.txt" dans le sous-dossier "logs"). Ensuite, il charge l'ensemble des sites comme expliqué ci-dessus, il ouvre le fichier contenant la base de données de trackers Ghostery et charge l'ensemble des expressions régulières utilisées afin de détecter les trackers. L'analyse des fichiers HTTP Archive commence et exporte pour chaque site analysé de multiples informations telles qu'expliqué dans la sous-section précédente. Lorsque tous les fichiers ont été analysés, le *parser* calcule les statistiques pour l'ensemble de l'analyse. Ensuite, il indique le nombre de fichiers ayant été correctement analysés et détaille les fichiers n'ayant pu être traités. Pour terminer, il ferme le fichier de log.

Chapitre 5

Résultats

Les expériences ont été réalisées avec Firefox 24.4.0 sur Linux à partir de différentes machines du pôle INGI [1] de l'UCL. Elles ont été lancées via un terminal et Xvfb [2] était préalablement lancé, cela permet d'éviter des interactions fortuites entre le navigateur et l'utilisateur. Les dernières versions des extensions *Firebug* et *NetExport* disponibles au moment de chaque expérience étaient installées (à l'exception de l'expérience à long terme où l'ensemble des composants de l'expérience est resté stable).

5.1 Expérience 1 : étude à long terme

5.1.1 Détails de l'expérience

Le but de cette expérience est de déterminer l'évolution des trackers sur une certaine période en effectuant des mesures de façon régulière.

Afin de réaliser cette étude, le *crawler* a été lancé chaque jour du 28 mars au 7 mai 2014 (à l'exception du 8 avril) sur le TOP 1000 du classement Alexa [42], ce qui représente 40 jours de mesures. Notez cependant que l'implémentation du compteur de cookies Flash a été réalisée ultérieurement à cette expérience et ces données ne sont donc pas disponibles au sein de l'expérience.

Le *crawler* a été lancé avec les paramètres suivants :

— les sites visités sont issus du TOP Alexa du 26 mars 2014
— l'intervalle sélectionné concerne les 1000 premiers sites
— 3 tentatives maximum ont été autorisées par site

1. Pôle d'ingénierie informatique
2. Xvfb est un serveur X qui permet d'effectuer les opérations graphiques en mémoire.

— Firefox était redémarré tous les 50 sites
— Firefox était dépourvu de toute extension (autre que Firebug et NetExport)

5.1.2 Premières constatations

Le graphique de la Figure 5.1 montre que l'efficacité du *crawler* reste relativement stable tout au long de l'expérience car on peut voir qu'au maximum 90 sites sur 1000 sont en échec, soit un taux d'échec toujours inférieur à 9%. Notez que l'implémentation finale du *crawler* a un peu changé depuis cette expérience. Lors de celle-ci, quand un site atteignait le nombre maximum de tentatives et qu'il était placé dans la liste des sites en échec, il était également retiré de la liste des sites ayant eu besoin d'au moins un nouvel essai suite à un timeout. Ceci explique des situations telles qu'au début du mois de mai où le nombre d'échecs est haut et le nombre de sites ayant été une fois en timeout est beaucoup plus bas.

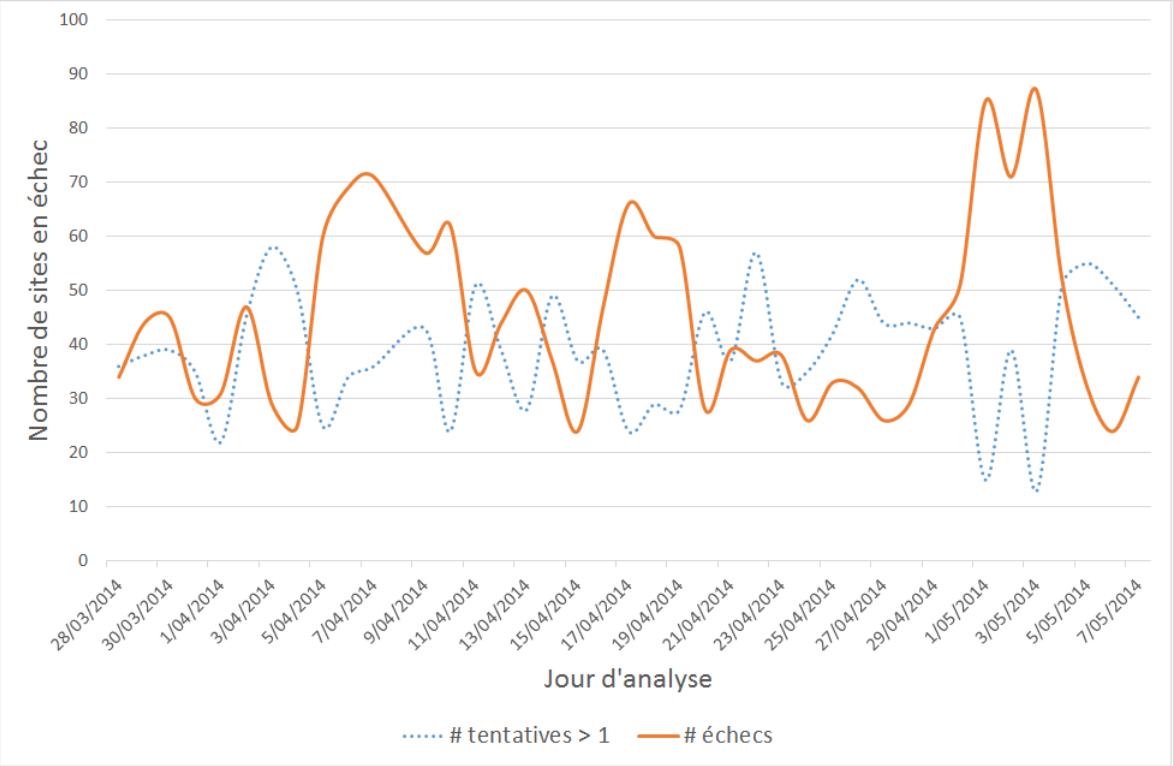

FIGURE 5.1 – Nombre de sites en échecs avec le *crawler*.

La deuxième partie de l'expérience consistait à lancer le *parser* afin d'analyser les fichiers générés lors du passage du *crawler* sur les sites. Le pourcentage de fichiers correctement analysés est représenté à la Figure 5.2. On peut voir que du début de l'expérience jusqu'au 21 avril, le pourcentage de réussite est en moyenne de 99,73%.

A partir du 22 avril et jusqu'à la fin de l'expérience, il passe à 98,71%, cela représente une dizaine de fichiers. Croyant d'abord à une erreur lors de l'exécution du *parser*, les analyses ont été reconduites une seconde fois mais n'ont pas montré de résultat différent. Après analyse des fichiers de log du *parser*, on remarque que ce sont généralement les mêmes sites qui sont en échec. Ainsi, sur les 16 jours (du 22 avril jusqu'au 7 mai), on peut remarquer certains faits communs entre les analyses : une série de sites a rencontré des problèmes en même temps alors que d'autres n'ont été touchés que de manière temporaire.

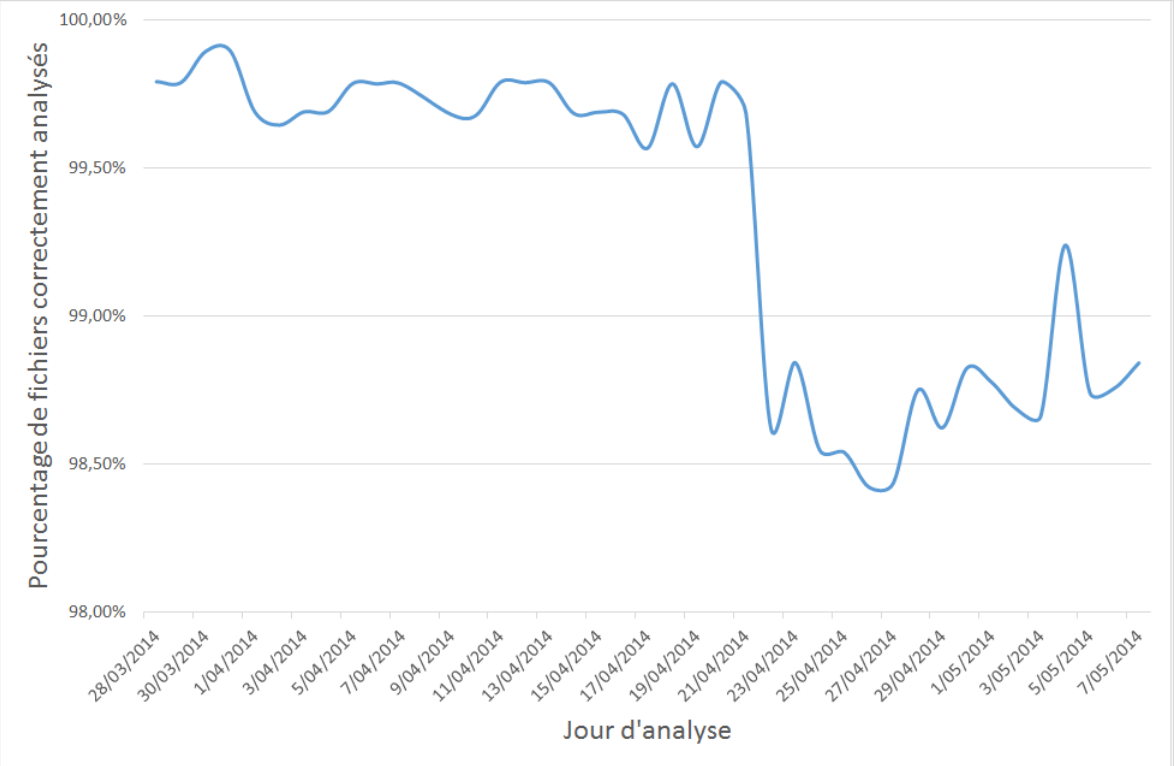

FIGURE 5.2 – Le pourcentage de réussite du *parser*.

La constatation que certains sites sont en échec pendant la période complète des 16 jours (alors qu'ils ne l'étaient pas avant le 22 avril) et que certains sites sont en échec sur une courte période (2-3 jours) fait penser que ces échecs résultent d'une erreur située sur les sites eux-mêmes.

Une recherche approfondie a été effectuée et celle-ci a confirmé l'hypothèse selon laquelle l'erreur provient des données reçues des serveurs. Le *parser* parcourt l'ensemble de chaque fichier HTTP Archive, celui-ci étant constitué de requêtes et réponses HTTP. La source des problèmes résulte d'une erreur dans certaines de ces réponses HTTP car elles ne respectent pas le standard défini par les RFC.

Plus précisément, les erreurs proviennent de champs non reconnus dans les réponses HTTP ou de valeurs erronées. Quelques exemples provenant des fichiers HTTP Archive ont été ajoutés afin d'illustrer les erreurs rencontrées.


```
"response": {
  "status": 302,
  "statusText": "Moved Temporarily",
  "httpVersion": "HTTP/1.1",
  "cookies": [
    {
      "name": "GAPS",
      "value": "1:G4yFvUco0ybt2VMj_gkGq4JfeZoaLw:Z3E-N608svwYh-H3",
      "path": "/",
      "expires": "2016-04-21T16:12:12.000+02:00",
      "secure": true,
      "httpOnly": true,
      "priority": "High",
      "domain": "accounts.google.com"
    }
  ],
```


FIGURE 5.3 – Réponse HTTP de *Google* et son cookie avec l'attribut "priority".

L'attribut *priority* n'existe pas dans l'entête *Set-Cookie* d'après la RFC 6265, section 5.2 [15]. Or, on peut voir dans la Figure 5.3 que Google émet une réponse HTTP avec un tel attribut pour la création d'un cookie. Etant donné que l'utilisation des services de Google est fortement répandue, cette erreur a été constatée dans plusieurs fichiers HTTP Archive.


```
"response": {
  "status": 200,
  "statusText": "OK",
  "httpVersion": "HTTP/1.1",
  "cookies": [
    {
      "name": "BAIDUID",
      "value": "54EA7793ABACAC88E873AF756E910EFE:FG",
      "expires": "NaN-NaN-NaNTNaN:NaN:NaN.NaN+NaN:NaN",
      "max-age": "31536000",
      "path": "/",
      "domian": ".baidu.com",
      "version": "1",
      "domain": "cb.baidu.com"
    }
  ],
```


FIGURE 5.4 – Réponse HTTP de *Baidu* avec un attribut "domian".

Dans la Figure 5.4, Baidu envoie une réponse HTTP qui contient une faute de frappe. En effet, au lieu de trouver un attribut "domain" (RFC 6265, section 5.2.3 [15]), on constate la présence d'un attribut "domian". De plus, un autre attribut "domain" est déjà présent.

```
"response": {
  "status": 301,
  "statusText": "Moved Permanently",
  "httpVersion": "HTTP/1.1",
  "cookies": [
    {
      "name": "vguid",
      "value": "f9622018-1acd-0ce1-fca6-737a5a44c020",
      "expire": "0",
      "path": "/",
      "domain": "gamespot.com"
    },
```

FIGURE 5.5 – Réponse HTTP avec la valeur "0" pour l'attribut "expire".

Dans l'exemple de la figure Figure 5.5, on constate également un problème : la présence d'un attribut "expire" dans l'entête *Set-Cookie* au lieu de "expires" (RFC 6265, section 5.2.1 [15]). De plus, le format de date dans cet attribut est invalide.

Ces différents exemples montrent que la source des problèmes de traitement des fichiers HTTP Archive ne provient pas de l'implémentation du *parser* mais provient plutôt d'erreurs dans les réponses HTTP reçues des serveurs.

5.1.3 Nombre de trackers

Le nombre total de trackers semble rester stable, il est en moyenne de 30080 (Figure 5.6). Ce nombre chute le 2 avril où il est de 25770 trackers. Ceci s'explique par le fait que le nombre de fichiers HTTP Archive pour ce jour (845 fichiers) est inférieur comparé aux autres jours, ce qui implique que moins de sites ont été analysés. Le nombre inférieur de fichiers provient probablement d'une perte de données suite à une suppression accidentelle ou à un manque d'espace disque qui a empêché l'enregistrement des fichiers HTTP Archive. L'export de ces fichiers étant externe à l'outil, celui-ci n'est pas en mesure de vérifier qu'ils sont correctement écrits sur le disque dur de la machine.

Des fluctuations dans le nombre total de trackers sont visibles car l'affichage des publicités sur les sites est généralement dynamique. Si une personne se rend sur une page puis la recharge, il est possible que les publicités soient différentes et proviennent d'autres régies publicitaires.

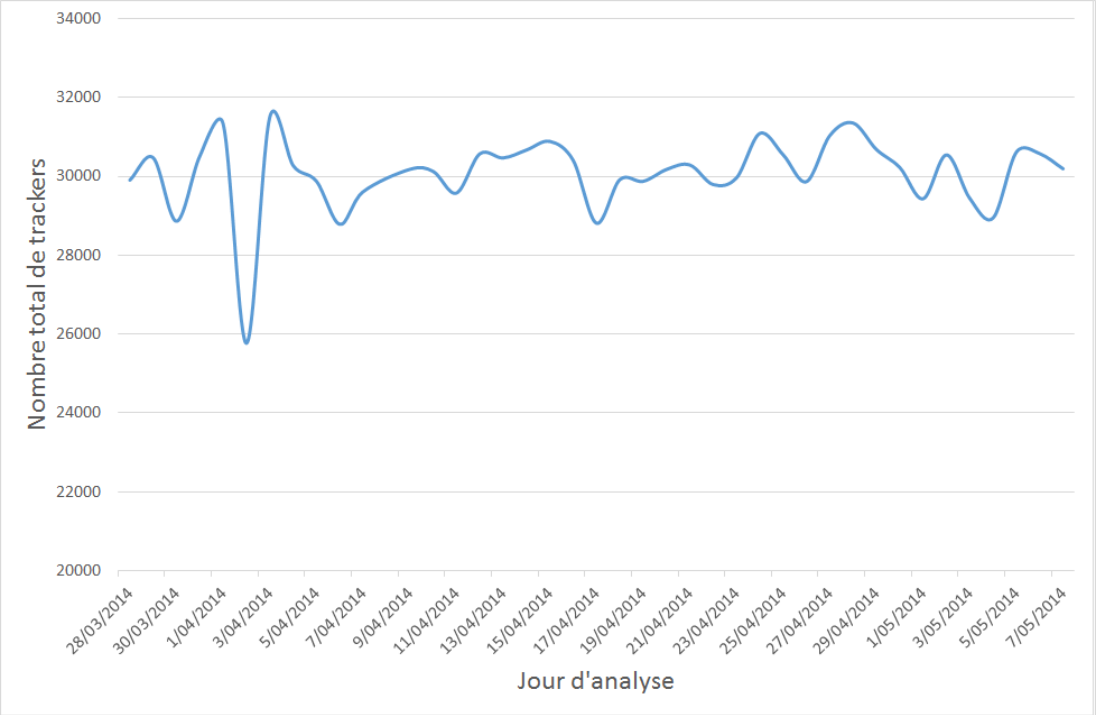

FIGURE 5.6 – Nombre total de trackers sur l'ensemble de l'analyse.

5.2 Expérience 2 : étude ponctuelle

Cette expérience a été lancée de manière ponctuelle afin d'établir une image des sites à un instant donnée. Elle permet aussi de constituer la base de référence pour la comparaison des extensions.

Le *crawler* a été lancé avec les paramètres suivants :

— les sites visités sont issus du TOP Alexa du 16 mai 2014
— l'intervalle sélectionné concerne les 1000 premiers sites
— 3 tentatives maximum ont été autorisées par site
— Firefox était redémarré tous les 50 sites
— Firefox était dépourvu de toute extension (autre que Firebug et NetExport)

Les 36 sites suivants ont été en échec, ce qui signifie qu'aucun fichier HTTP Archive n'a été généré pour eux :

— 39.net
— aili.com
— baomihua.com
— bitauto.com
— caijing.com.cn
— china.com
— csdn.net
— enet.com.cn
— gazeta.ru
— gmw.cn
— hao123.com
— heise.de
— howstuffworks.com
— ifeng.com
— ileehoo.com

- justdial.com
- kdnet.net
- lady8844.com
- microsoftonline.com
- moz.com
- online.sh.cn
- onlinesbi.com
- opensiteexplorer.org
- pchome.net
- people.com.cn
- pinimg.com
- qq.com
- secureserver.net
- sina.com.cn
- sohu.com
- tudou.com
- twimg.com
- yesky.com
- yoka.com
- youku.com
- zing.vn

Ces sites ne seront par conséquent pas analysés dans cette expérience.

Par ailleurs, 78 sites ont subi au moins un timeout mais ils ne seront pas listés ici. Notez que si ces sites ont rencontré 3 timeouts consécutifs, ils sont présents dans la liste des sites en échec (ci-dessus).

Le *parser* a été lancé avec le paramètre suivant :
- la base de données Ghostery utilisée est la version 303

Le *parser* n'a pas été en mesure d'analyser les fichiers suivants (les causes sont identiques à celles de l'expérience à long terme) :

- 9gag.com.har
- accounts.google.com-2.har
- mashable.com.har
- website-unavailable.com-1.har
- www.babytree.com.har
- www.deezer.com.har
- www.gamespot.com.har
- www.gmail.com.har
- www.pixiv.net.har
- www.thefreedictionary.com.har
- www.theverge.com-1.har
- www.wikihow.com.har
- www.wix.com.har

Les sites correspondants à ces fichiers ne seront par conséquent pas analysés dans cette expérience.

L'analyse des fichiers HTTP Archive par le *parser* a été effectuée deux fois : la première avec la base de données Ghostery et la seconde fois, sans cette base de données.

5.2.1 Carte des trackers détectés

La répartition des trackers par catégorie et pour les deux analyses est disponible aux Figures 5.7 et 5.9.

Concernant les cookies Flash, 79 sites ont créé au moins un fichier *.sol*. Le site en ayant créé le plus (10 fichiers .sol) est *espncricinfo.com*. En deuxième position, on trouve *espn.go.com* avec 8 fichiers .sol créés lors de la visite du site. Les deux sites suivants sont *nordstrom.com* et *loading-delivery1.com* avec 4 fichiers .sol créés. Les 75 autres sites ont créé 3 cookies (9 sites), 2 cookies (9 sites) et 1 cookie (57 sites).

Analyse avec la base de données Ghostery

Comme on peut le constater sur la Figure 5.7, la majorité des trackers détectés le sont par Ghostery (84%). Ceci est normal car un premier tri est effectué avec l'aide de la base de données Ghostery. En deuxième position, viennent les URL externes appelées avec des paramètres (1350, soit 4,59%) et en troisième position, les pixels de traçage de domaines tiers. Il apparaît qu'une partie non négligeable des pixels de traçage n'est pas détectée par Ghostery. Leur nombre est de 1212, ce qui représente 4,12% de l'ensemble des trackers détectés par l'outil. Les codes JavaScript appelés avec des paramètres ont été détectés 1079 fois (3,67%). Il y a également 888 réponses HTTP provenant de sites externes (3,02%) qui créent un cookie tiers. Enfin, peu de fichiers Flash sont détectés par l'outil car ils représentent 0,84% des éléments détectés, soit 246 éléments. Au total, 29402 trackers ont donc été détectés.

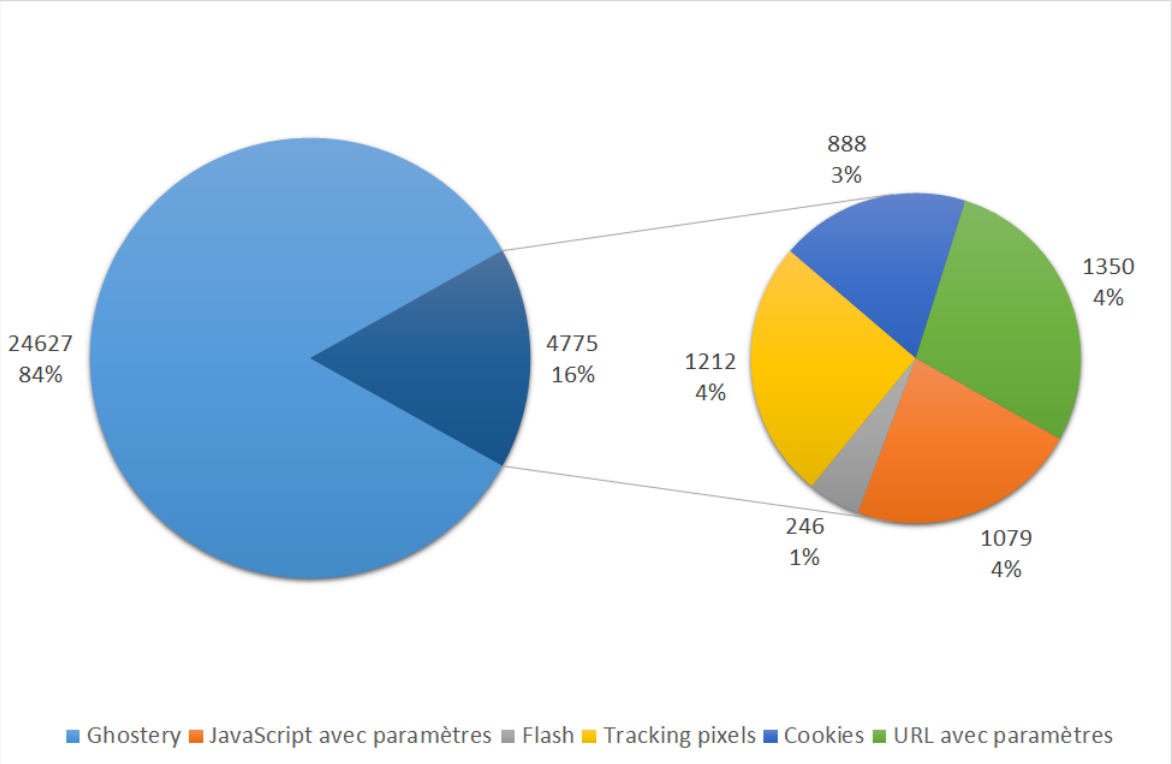

FIGURE 5.7 – Répartition des trackers par catégorie (avec Ghostery).

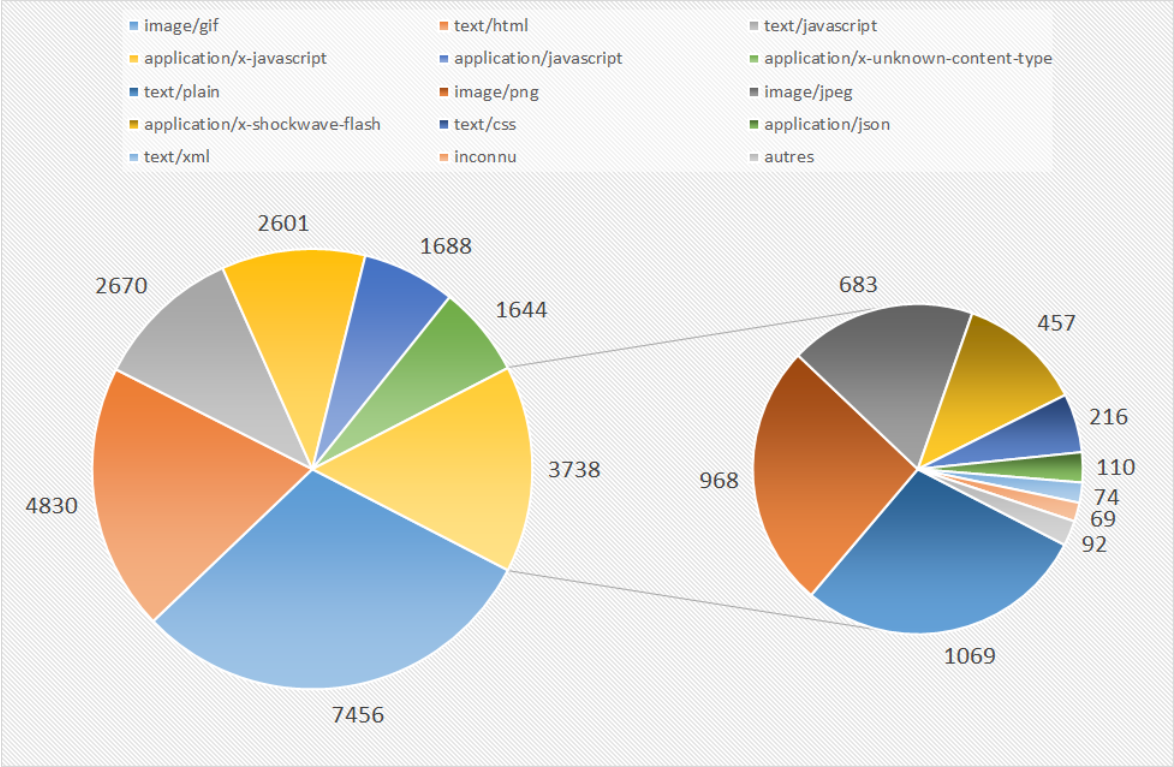

FIGURE 5.8 – Répartition des types de trackers détectés par Ghostery.

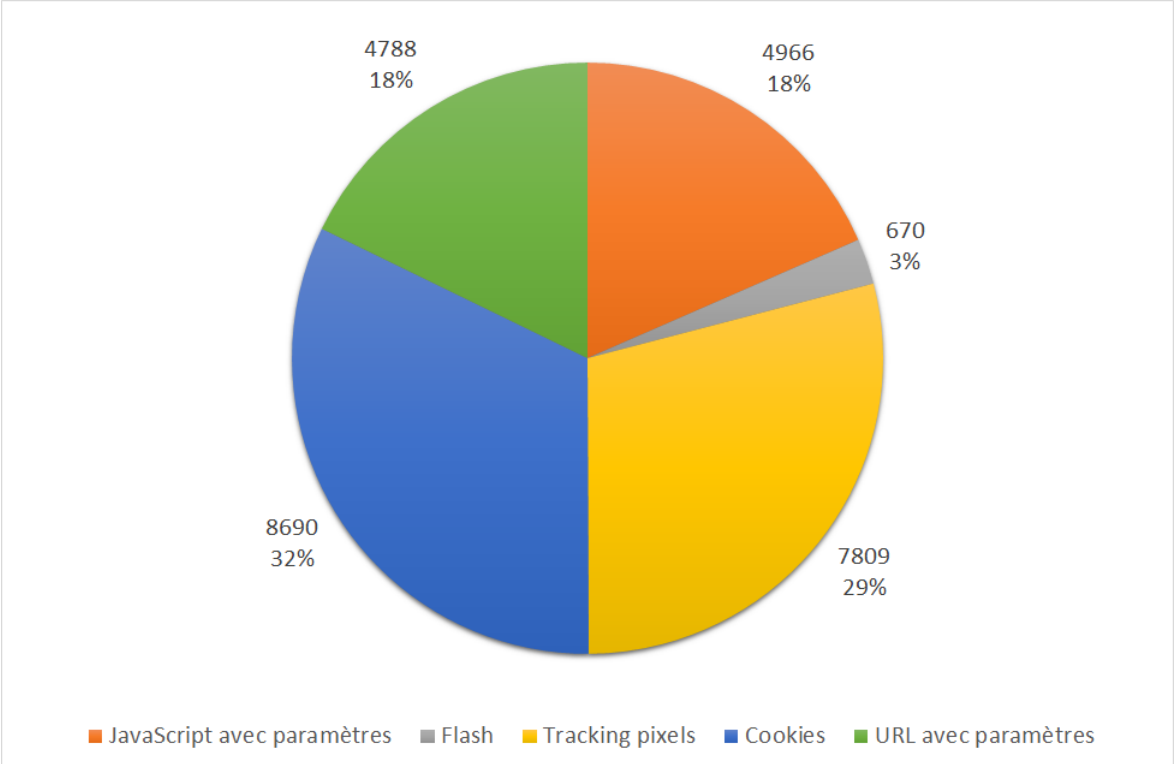

FIGURE 5.9 – Répartition des trackers par catégorie (sans Ghostery).

Lorsqu'une ressource est détectée comme étant un tracker par Ghostery, le *parser* enregistre le type de cette ressource (voir des exemples dans l'Annexe D). Les résultats sont visibles à la Figure 5.8. Ils montrent que le premier type de trackers est de type *.gif* (7456 trackers). La deuxième position est occupée par le JavaScript (si on rassemble *text/javascript*, *application/x-javascript* et *application/javascript*) avec 6959 trackers et la troisième position concerne les ressources de type "text/html".

Ceci montre que les principaux trackers importés de sites tiers sont les pixels de traçage ainsi que les codes écrits en JavaScript.

Au vu des deux graphiques (Figures 5.7 et 5.8), l'origine principale des trackers est la présence d'images *.gif* jouant le rôle de pixels de tracking et de l'inclusion de codes JavaScript. Ces trackers représentent la majorité des éléments détectés par Ghostery et ils sont respectivement au nombre de 1212 et 1079 d'après les critères opérant après le premier tri effectué par Ghostery.

Analyse sans la base de données Ghostery

La Figure 5.9 reprend l'ensemble des trackers détectés lors de l'analyse effectuée sans la prise en compte de la base de données Ghostery. Pour rappel, seuls les éléments chargés depuis un domaine tiers sont présents dans ce graphique.

La plus grande portion du graphique (32,28%) se compose des créations de cookies tiers. La seconde (29%) est constituée des pixels de traçage et le trio de tête se termine par les codes JavaScript chargés avec des paramètres (18,45%). Les URL appelées avec des paramètres sont juste derrière avec 17,78% des trackers détectés et les fichiers Flash représentent (2,49%). Au total, 26923 trackers ont été repérés.

5.2.2 Discussion sur les résultats des deux types d'analyse

On remarque que le nombre de trackers détectés est plus élevé pour l'analyse ayant utilisé la base de données Ghostery (29402 contre 26923). Ceci s'explique par plusieurs raisons.

La première est que les images issues de domaines tiers et connues comme étant des trackers sont détectées par Ghostery mais ne le sont pas par les autres critères. Une telle détection n'est pas envisageable sans l'aide d'une base de données car il est impossible de savoir ce qu'il se passe du côté du serveur. Par contre, les pixels espions sont détectés grâce à leur particularité.

La seconde raison porte sur les codes JavaScript. Seuls les codes JavaScript appelés avec une requête HTTP dont l'URI possède une chaîne de requête sont considérés comme des trackers par les critères définis. Il existe également des trackers JavaScript qui n'ont pas de chaîne de requête mais les considérer systématiquement comme des trackers provoquerait beaucoup de faux positifs. La solution idéale serait d'analyser le comportement de ces scripts afin de déterminer si ce sont des trackers ou non.

A l'inverse, le nombre de trackers détectés par l'analyse n'utilisant pas la base de données Ghostery peut parfois être supérieur dans certaines catégories. En effet, les critères retiennent toutes les ressources Flash et les URL possédant un champ de requête. Cependant, tous ces éléments ne sont pas des trackers même si force est de constater que la plupart en sont.

L'utilisation d'une base de données permet de détecter des éléments qui ne seraient pas suspectés d'être des trackers. Cependant, elle peut également passer à côté de trackers qui ne sont pas renseignés dans cette base.

Cela est d'ailleurs flagrant avec la détection des pixels espions. Malgré l'utilisation de la base de données Ghostery, 1212 pixels de traçage ont été détectés par les critères définis au sein du *parser*.

Chaque analyse a ses avantages et ses inconvénients : celle qui n'utiliserait que la base de données Ghostery aurait peu (voire pas) de faux positifs mais passe à côté de certains trackers absents de sa base de données alors que l'analyse n'utilisant pas cette base peut renfermer plus de faux positifs mais détecte aussi plus de trackers de certaines catégories.

L'outil développé pour ce mémoire propose deux types d'analyses : le premier utilise la base de données Ghostery et applique les critères définis sur les ressources n'ayant pas été détectées par la base de données. Le second ne repose que sur les critères définis.

Un expérimentateur choisissant le premier type d'analyse bénéficiera donc des avantages de l'utilisation de la base de données mais également des critères définis.

5.2.3 Sites renfermant le plus de trackers détectés

Le TOP 10 des sites en nombre de trackers détectés pour les deux analyses sont :

Rang	Avec la base Ghostery	#	Sans la base Ghostery	#
1	drudgereport.com	341	drudgereport.com	385
2	www.hongkiat.com	333	www.primewire.ag	367
3	www.primewire.ag	284	www.hongkiat.com	364
4	www.theblaze.com	256	www.theblaze.com	332
5	photobucket.com	240	photobucket.com	280
6	slickdeals.net	226	www.jeuxvideo.com	271
7	www.linternaute.com	223	tinypic.com	268
8	www.jeuxvideo.com	213	slickdeals.net	248
9	www.engadget.com	209	www.linternaute.com	240
10	tinypic.com	201	www.commentcamarche.net	235

Les deux analyses rapportent des résultats similaires pour la désignation des 10 sites renfermant le plus de trackers à l'exception de *www.engadget.com* qui se trouve seulement dans le TOP 10 de l'analyse utilisant la base de données Ghostery et *www.commentcamarche.net* qui se trouve seulement dans le TOP 10 de l'autre analyse qui n'utilise pas la base de données de trackers.

5.2.4 Organisations déployant le plus de trackers

Selon cette analyse, en ne considérant que les trackers détectés par Ghostery, voici la liste des organisations qui en génèrent le plus.
Sur un total de 24626 trackers :

- 2562 : DoubleClick
- 1379 : AppNexus
- 949 : Google Analytics
- 675 : Google Adsense
- 574 : Rubicon
- 534 : Turn
- 441 : ScoreCard Research Beacon
- 347 : MediaMath
- 332 : Lotame
- 326 : OpenX

Quand on sait que *DoubleClick* appartient à *Google*, cela fait 4856 trackers provenant du géant mondial. Cela représente un peu moins de 20% des trackers détectés au total par la base de données Ghostery. Le second est *AppNexus* avec 5,60% (1379 trackers), il s'agit d'une plateforme fonctionnant sous le modèle RDB [3].

3. RDB (Real Time Bidding) est un système qui permet à des annonceurs de placer leurs publicités selon un système d'enchères. Il repose sur une plateforme d'achat et vente de publicités en temps réel.

Ensuite, on trouve Facebook et Adobe quasiment ex æquo avec respectivement 2,81% (691 trackers) et 2,78% (685 trackers). Pour terminer, il y a 2 autres entreprises au dessus de la barre des 2%, 7 entreprises entre 1 et 2% et les autres (plus de 500 entreprises) avec un pourcentage inférieur à 1%.

Il faut également noter qu'AppNexus utilise différents services dont principalement ceux de Google (en 2010, il semblerait que Google comptait pour 50% [4]). Il est donc possible que cela permette à Google d'amasser encore plus de statistiques grâce à cet intermédiaire. Cette répartition montre que de grands groupes tracent massivement les utilisateurs mais qu'il existe énormément de petites entreprises qui tracent également les utilisateurs mais dans une moindre mesure.

4. D'après un expert de l'industrie publicitaire (`http://techcrunch.com/2010/11/30/google-temporarily-blocks-appnexus-from-its-ad-exchange/`)

Chapitre 6

Moyens de défense

Grâce à l'outil réalisé, nous sommes en mesure de tester l'efficacité de différents moyens de défense. Le but de ce chapitre est donc de déterminer quelle est la meilleure protection disponible dans Firefox contre les différents types de trackers détectables par l'outil.

6.1 Extensions des navigateurs

Certaines extensions de navigateurs ont été développées afin de préserver la vie privée des utilisateurs. Dans cette section, plusieurs de ces extensions disponibles pour Firefox vont être testées selon une expérience ponctuelle (voir section 5.2).

Le *crawler* a chaque fois été configuré pour analyser le TOP 1000 du classement Alexa [42] du 16 mai 2014. La méthodologie pour la préparation de chaque analyse est la suivante :

— chaque profil Firefox est créé expressément pour chaque analyse
— les extensions Firebug et NetExport sont installées
— l'extension NetExport est modifiée (voir section 4.2.1)
— l'extension devant être testée est installée et configurée

Le *crawler* génère ainsi les fichiers HTTP Archive et détermine le nombre de cookies Flash créés par chaque site.

Pour terminer, le *parser* traite les fichiers HTTP Archive afin de fournir les données essentielles à l'élaboration des résultats. Le *parser* a utilisé la version 303 de la base de données de trackers Ghostery.

Le but de chaque expérience est de déterminer si le nombre de trackers détectés par le *parser* est effectivement en baisse grâce aux extensions installées. L'ensemble de ces différentes expériences permettra ensuite de déterminer quelle protection semble la meilleure.

Pour chaque expérience, seules les analyses utilisant la base de données Ghostery ont été réalisées. Suite à la discussion portant sur les deux types d'analyses (sous-section 5.2.2), il semble que les analyses utilisant les données issues de Ghostery soient les plus pertinentes pour les tests d'extensions de Firefox.

Les résultats sont toujours présentés dans un même modèle de tableau :

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery et critères triés			Pourcentage
⋮	en ordre décroissant par	⋮	⋮	par rapport
6	rapport au nombre de trackers			à la référence
	TOTAL		-	

Notez que la somme des pourcentages de répartition des trackers dans le tableau peut ne pas être exactement de 100% à cause des arrondis.

Afin de ne pas surcharger ce chapitre de graphiques, les types MIME des éléments détectés par la base de données Ghostery n'ont pas été inclus mais leurs résultats sont énoncés dans les analyses de chaque expérience. Les types *application/x-javascript*, *application/javascript* et *text/javascript* ont été rassemblés sous le nom *JavaScript* car ils représentent le même type de trackers. Ils sont présentés sous cette forme :

Rang	Types MIME (Ghostery)	#	%	Évolution
1	Types MIME triés			Pourcentage
⋮	en ordre décroissant	⋮	⋮	par rapport
5				à la référence

Étant donné que seuls les 5 premiers types MIME sont présents dans le tableau, il est normal que la somme des pourcentages n'atteigne pas 100%.

6.1.1 Données de référence

Pour rappel, voici les données de référence du chapitre précédent (Figures 5.7 et 5.8), mises sous forme de tableaux :

Rang	Répartition des trackers	#	%
1	Ghostery	24627	83,76
2	URL avec paramètres	1350	4,59
3	Tracking pixels	1212	4,12
4	JavaScript avec paramètres	1079	3,67
5	Cookies	888	3,02
6	Flash	246	0,84
	TOTAL	29402	

Rang	Types MIME (Ghostery)	#	%
1	Images .gif	7456	30,28
2	JavaScript	6959	28,26
3	HTML	4830	19,61
4	Type inconnu	1644	6,68
5	Texte	1069	4,34

6.1.2 Adblock Plus

Présentation

Adblock Plus [1] est une extension disponible pour Firefox, Google Chrome, Opera, Safari, Internet Explorer et Android. Son fonctionnement repose sur des filtres installés au choix par l'utilisateur. Le but d'Adblock Plus est de bloquer les publicités intrusives et n'autoriser que les publicités jugées acceptables. En effet, les sites dont les publicités sont considérées comme acceptables (des critères stricts ont été définis par Adblock Plus) peuvent demander à être intégrés dans une liste d'exceptions afin de rendre leurs publicités visibles par les utilisateurs. Cependant, l'utilisateur est libre d'afficher ces publicités. De plus, il est possible de créer des listes personnalisées.

Adblock propose des listes par défaut régulièrement mises à jour. Certaines concernent les publicités indésirables de façon générale et d'autres sont spécifiques à des langues. Plusieurs miroirs existent et proposent de bloquer des types précis

1. `https://adblockplus.org/`

d'éléments (modules de réseaux sociaux par exemple). Un de ces miroirs connus est Fanboy [2], il propose plusieurs listes dont certaines bloquent des éléments qui tracent les utilisateurs. Les listes ont généralement une durée de vie de 4 jours.

Configuration de l'extension

Étant donné que les résultats peuvent varier en fonction des filtres installés, trois expériences ont été réalisées avec cette extension.

La première utilise les paramètres par défaut : le filtre de base (Easylist) et un filtre basé sur la langue.

La seconde expérience est basée sur les mêmes réglages que la première mais bloque également les publicités dites acceptables par Adblock.

La troisième utilise un filtre de blocage plus complet. Il s'agit de la liste "Ultimate" Fanboy [3], cette liste inclut les éléments suivants : Easylist, Easyprivacy, Enhanced Trackers List and Annoyances List.

Notez que des profils Firefox différents ont été configurés pour réaliser les différentes expériences afin d'éviter une quelconque influence entre elles.

Résultats de la première expérience - Figure 6.1

L'extension a laissé passer 2584 trackers connus de la base de données Ghostery et 1631 éléments ont été identifiés comme trackers selon les critères de l'outil.

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery	2584	61,30	- 89,51%
2	URL avec paramètres	563	13,36	- 58,30%
3	JavaScript avec paramètres	558	13,24	- 48,29%
4	Cookies	303	7,19	- 65,88%
5	Flash	118	2,80	- 52,03%
6	Tracking pixels	89	2,11	- 92,66%
	TOTAL	4215	-	- 85,66%

2. `https://www.fanboy.co.nz/`
3. `http://www.fanboy.co.nz/filters.html`

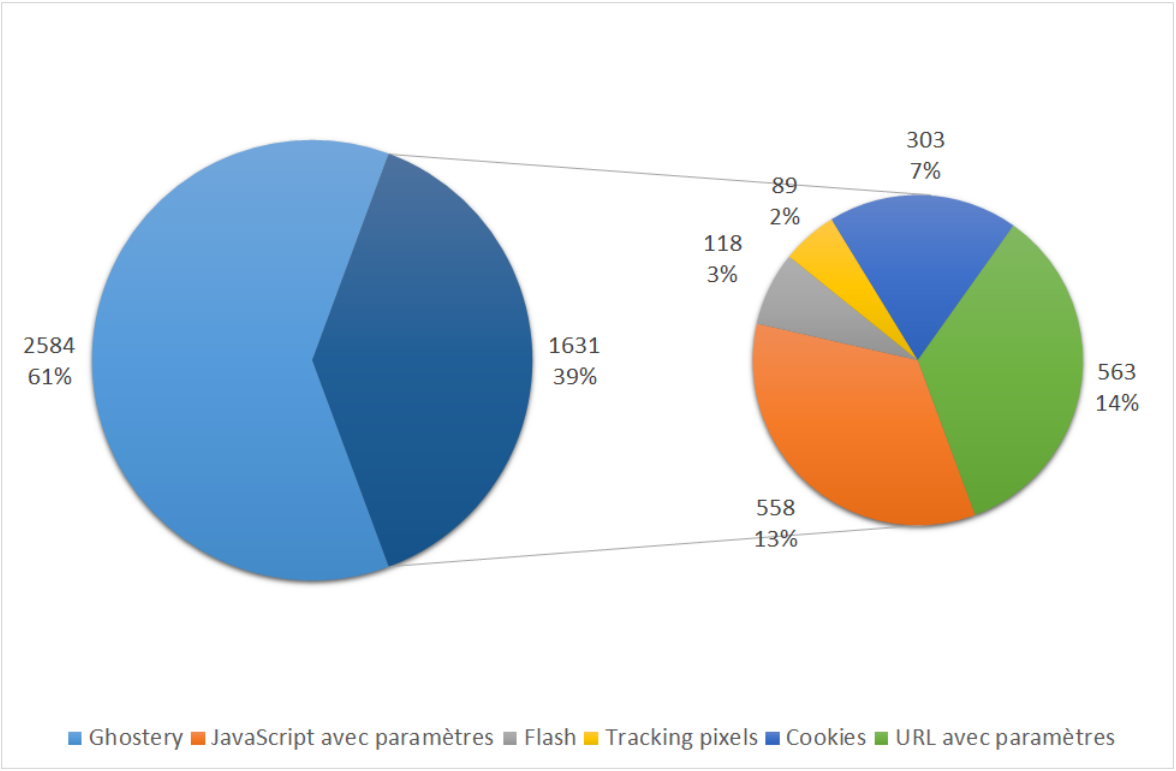

FIGURE 6.1 – Répartition des trackers par catégorie.

Rang	Types MIME (Ghostery)	#	%	Évolution
1	JavaScript	1136	43,96	- 83,68%
2	Images .png	396	15,33	- 59,09%
3	HTML	322	12,46	- 93,33%
4	Images .jpeg	300	11,61	- 56,08%
5	Images .gif	151	5,84	-97,97%

Deux baisses flagrantes sont constatées pour les pixels de traçage et les trackers connus de Ghostery. Dans ces derniers, ce sont principalement les images .gif, les requêtes de ressources HTML et les JavaScript qui sont bloqués par Adblock.

Résultats de la seconde expérience - Figure 6.2

L'extension a laissé passer 2572 trackers connus de la base de données Ghostery et 1555 éléments ont été identifiés comme trackers selon les critères de l'outil.

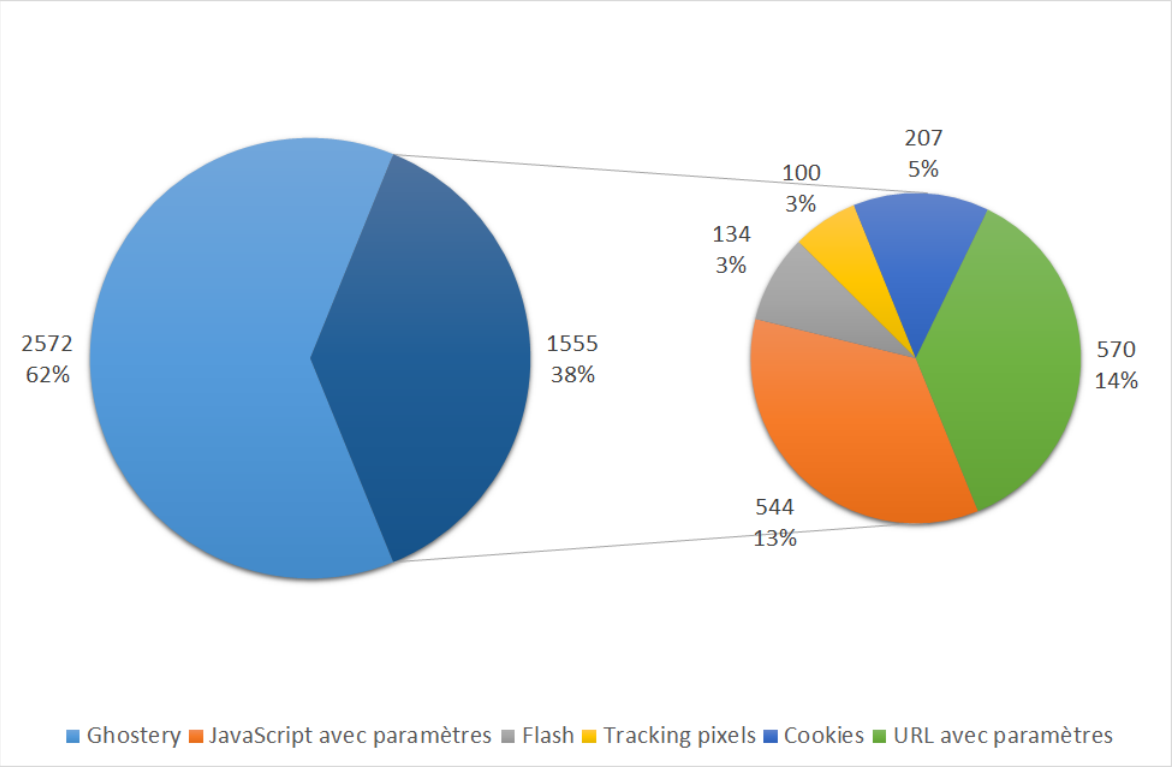

FIGURE 6.2 – Répartition des trackers par catégorie.

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery	2572	62,32	- 89,56%
2	URL avec paramètres	570	13,81	- 57,78%
3	JavaScript avec paramètres	544	13,18	- 49,58%
4	Cookies	207	5,02	- 76,69%
5	Flash	134	3,25	- 45,53%
6	Tracking pixels	100	2,42	- 91,75%
	TOTAL	4127	-	- 85,96%

Rang	Types MIME (Ghostery)	#	%	Évolution
1	JavaScript	1134	44,09	- 83,70%
2	Images .png	407	15,82	- 57,95%
4	HTML	330	12,83	- 93,17%
4	Images .jpeg	245	9,53	- 64,13%
5	Images .gif	161	6,26	- 97,84%

Le fait de bloquer les publicités dites acceptables par les développeurs d'Adblock fait baisser un peu plus le niveau de trackers détectés : on passe de 4215 à 4127

trackers. Cette baisse est essentiellement causée par la diminution du nombre de réponses HTTP créant un cookie tiers (on diminue de 300 à 207). On constate également un nombre légèrement supérieur pour les ressources Flash, les pixels de traçage et les URL avec paramètres. Après vérification du log du *parser*, 4 fichiers HTTP Archive ont été analysés en plus, ce qui provoque probablement ces légères hausses.

Résultats de la troisième expérience - Figure 6.3

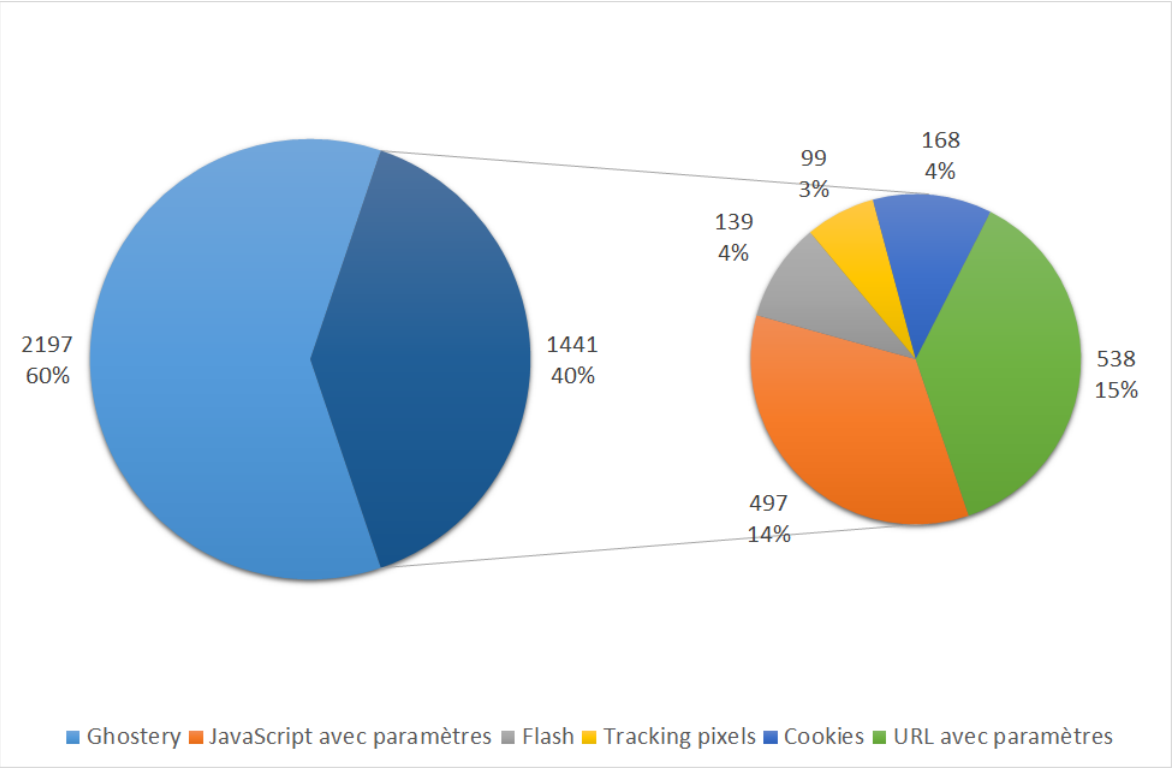

FIGURE 6.3 – Répartition des trackers par catégorie.

L'extension a laissé passer 2197 trackers connus de la base de données Ghostery et 1441 éléments ont été identifiés comme trackers selon les critères de l'outil.

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery	2197	60,39	- 91,08%
2	URL avec paramètres	538	14,79	- 60,15%
3	JavaScript avec paramètres	497	13,66	- 53,94%
4	Cookies	168	4,62	- 81,08%
5	Flash	139	3,82	- 43,50%
6	Tracking pixels	99	2,72	- 91,83%
	TOTAL	3638	-	- 87,63%

Rang	Types MIME (Ghostery)	#	%	Évolution
1	JavaScript	764	34,77	- 89,02%
2	Images .png	373	16,98	- 61,47%
3	HTML	318	14,47	- 93,42%
4	Images .jpeg	316	14,38	- 53,73%
5	Images .gif	152	6,92	- 97,96%

L'utilisation de la liste Fanboy Ultimate fait baisser davantage le nombre de trackers car on passe de 4127 pour l'expérience précédente (4215 pour la configuration par défaut d'Adblock) à 3638 trackers détectés. Pour les trackers de la base de données Ghostery, la baisse vient principalement du JavaScript avec une baisse de 1134 (1136 pour la configuration par défaut d'Adblock) à 764 codes JavaScript connus comme trackers ayant été détectés. Concernant les autres critères, une légère baisse des URL et JavaScript avec des paramètres est visible ainsi que pour les cookies.

6.1.3 DoNotTrackMe

Présentation

DoNotTrackMe [4] est une extension disponible pour Google Chrome, Firefox, Safari, Opera et Internet Explorer. Elle bloque les trackers de différents types de sociétés (régies publicitaires, réseaux sociaux et sociétés qui récupèrent des données sur la navigation des utilisateurs). Il est expliqué sur le site de l'extension que son but n'est pas de bloquer toutes les publicités mais plutôt de bloquer les publicités ciblées qui utilisent des informations personnelles. Son fonctionnement repose sur une base de données de trackers. A chaque page ouverte dans le navigateur, l'extension vérifie les ressources qui sont chargées et bloque celles qui sont connues comme trackers.

4. `https://www.abine.com/donottrackme.html`

Configuration de l'extension

L'extension n'a pas été configurée, les paramètres par défaut ont été laissés d'application. A vrai dire, les options de configuration sont assez rudimentaires. L'extension possède un écran de contrôle qui compte le nombre de trackers bloqués. Un compte premium est également proposé mais il n'est pas intéressant vu les fonctionnalités recherchées au sein de l'expérience.

Résultats - Figure 6.4

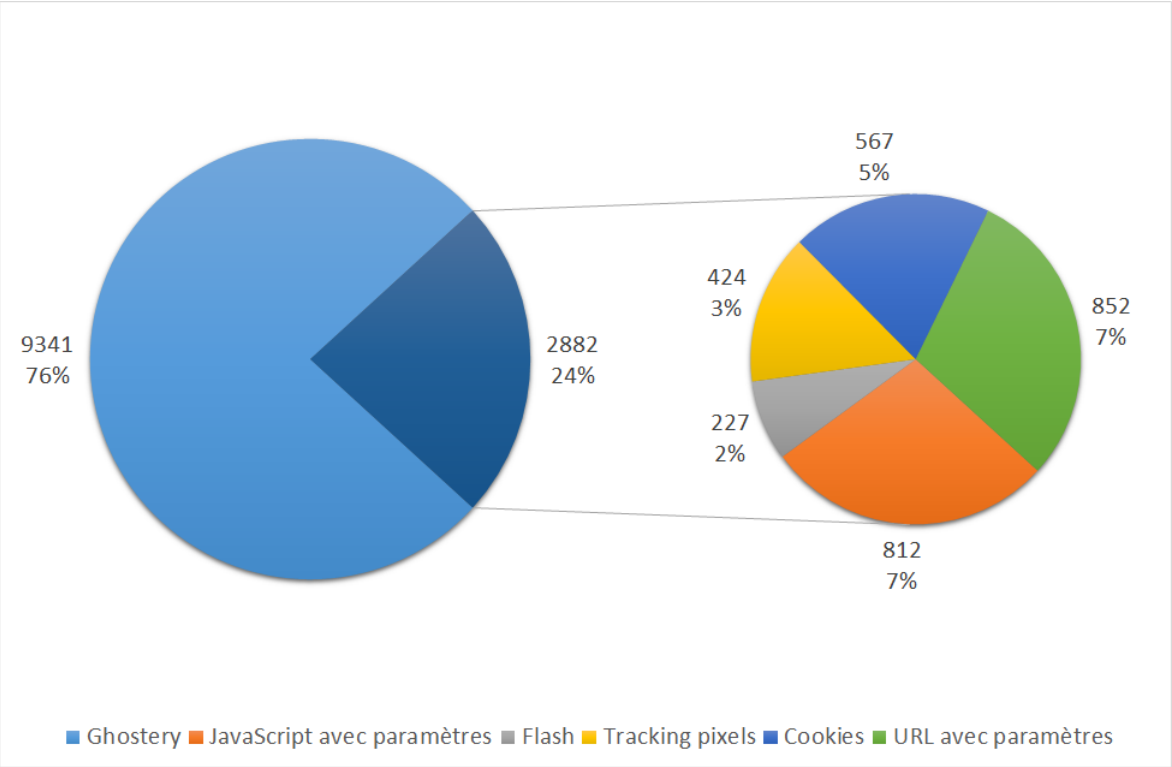

FIGURE 6.4 – Répartition des trackers par catégorie.

L'extension a laissé passer 9341 trackers connus de la base de données Ghostery et 2882 éléments ont été identifiés comme trackers selon les critères de l'outil.

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery	9341	76,42	- 62,07%
2	URL avec paramètres	852	6,97	- 36,89%
3	JavaScript avec paramètres	812	6,64	- 24,75%
4	Cookies	567	4,64	- 36,15%
5	Tracking pixels	424	3,47	- 65,02%
6	Flash	227	1,86	- 7,72%
	TOTAL	12223	-	- 58,43%

Rang	Types MIME (Ghostery)	#	%	Évolution
1	JavaScript	3441	36,84	- 50,55%
2	Images .gif	2124	22,74	- 71,51%
3	HTML	1400	14,99	- 71,01%
4	Type inconnu	529	5,66	- 67,82%
5	Images .png	517	5,53	- 46,59%

L'extension offre une protection relativement moyenne car elle ne bloque que 58,43% des trackers par rapport à l'expérience de référence. Elle bloque cependant 65,02% de pixels de traçage et 62,07% des trackers connus par Ghostery. Dans ces derniers, ce sont les pixels de traçage (71,51%) et les URL avec une chaîne de requête qui sont proportionnellement les plus bloqués (71,01% pour le type HTML et 67,82% pour les types inconnus). Quant aux codes JavaScript, ils sont en tête trackers bloqués avec un nombre de 3441.

6.1.4 Ghostery

Présentation

Ghostery [5] est une extension disponible pour Firefox, Google Chrome, Opera et Safari. Son fonctionnement repose sur une base de données de trackers alimentée par les retours des utilisateurs de l'extension. Cette base de données est régulièrement mise à jour. Evidon, la société qui possède Ghostery, a fait momentanément parler d'elle car elle a des contrats avec des entreprises du domaine publicitaire. Certains lui reprochaient alors de vendre les informations reçues des utilisateurs via son programme Ghostrank. Sur leur site, les développeurs de Ghostery assurent ne pas vendre d'informations personnelles. Il est possible de désactiver Ghostrank.

5. `https://www.ghostery.com`

Configuration de l'extension

L'extension a été activée avec tous les trackers et tous les cookies sélectionnés.

Résultats - Figure 6.5

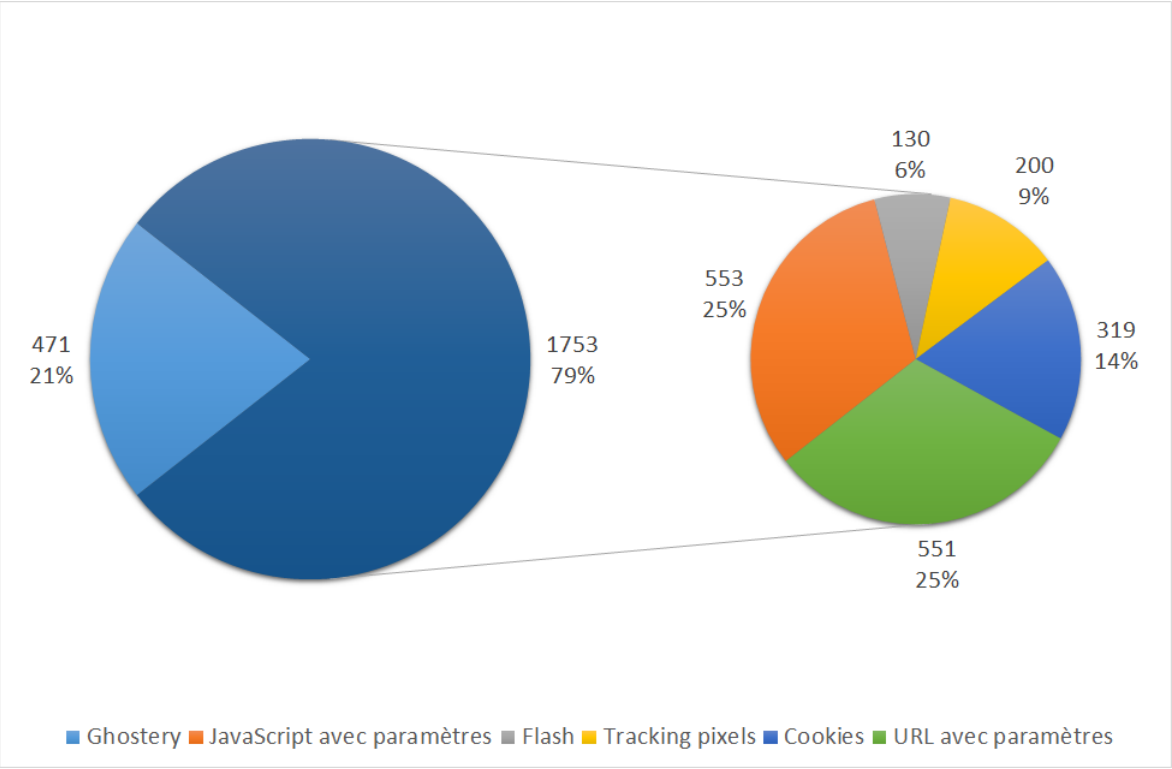

FIGURE 6.5 – Répartition des trackers par catégorie.

L'extension a laissé passer 471 trackers connus de la base de données Ghostery et 1753 éléments ont été identifiés comme trackers selon les critères de l'outil.

Rang	Répartition des trackers	#	%	Évolution
1	JavaScript avec paramètres	553	14,53	- 48,75%
2	URL avec paramètres	551	14,48	- 59,19%
3	Ghostery	471	12,38	- 98,09%
4	Cookies	319	8,38	- 64,08%
5	Tracking pixels	200	5,25	- 83,50%
6	Flash	130	3,42	- 47,15%
	TOTAL	2224	-	- 92,44%

Rang	Types MIME (Ghostery)	#	%	Évolution
1	Images .png	111	23,57	- 88,53%
2	Images .jpeg	92	19,53	- 86,53%
3	JavaScript	86	18,26	- 98,76%
4	Images .gif	62	13,16	- 99,17%
5	HTML	56	11,89	- 98,84%

Les résultats de l'analyse ont été assez étonnants car ils montrent qu'une partie des trackers n'est pas bloquée par Ghostery alors qu'ils sont bien présents dans la base de données. Ils sont détectés par le *parser* qui utilise la même base de données. Une analyse plus poussée sur les liens non bloqués a été effectuée. Le constat est que Ghostery ne bloque pas les trackers du site visité (les trackers "first-party"). Après une recherche, la justification a été trouvé sur un billet de blog [6] des développeurs de l'extension. La raison principale est que bloquer ces trackers peut entraîner des interférences lors de l'utilisation des fonctionnalités du site.

Néanmoins, l'extension offre une bonne protection car elle bloque 92,44% des trackers. Les principaux trackers non bloqués sur les sites visités sont les images .png et .jpeg (respectivement 23,57% et 19,53%), suivies des JavaScript (18,26%) et des images .gif (13,16%). Concernant les autres critères, les pixels de traçage sont les mieux bloqués avec un taux de 83,50%, cela signifie que 200 pixels passent quand même à côté de la base de données Ghostery.

6.1.5 HTTPS Everywhere

Présentation

HTTPS Everywhere [7] est une extension disponible pour Firefox, Google Chrome, Opera et Android. Firefox dispose d'une version stable tandis que les autres n'ont accès qu'à une version bêta. Le but de cette extension est d'utiliser les connexions HTTPS dès que cela est possible. L'extension repose sur une liste afin de savoir quand passer d'une connexion HTTP vers une connexion HTTPS. Par défaut, l'extension contient une liste de règles prédéfinies pour chaque site mais il est également possible de créer ses propres règles. Grâce à l'utilisation de connections sécurisées, l'extension protège l'utilisateur de certaines attaques. Dès qu'une connexion sécurisée peut être établie vers un site, HTTPS Everywhere le fait. Il peut cependant

6. `https://purplebox.ghostery.com/post/1016021484`
7. `https://www.eff.org/https-everywhere`

arriver que certaines requêtes vers un site soient en HTTP alors que d'autres sont en HTTPS. L'extension n'est pas en mesure de protéger l'utilisateur si le site ne propose pas de connexion HTTPS pour toutes ses ressources.

Configuration de l'extension

L'extension a été installée avec ses paramètres par défaut. SSL Observatory n'a pas été activé et aucune règle supplémentaire n'a été ajoutée.

Résultats - Figure 6.6

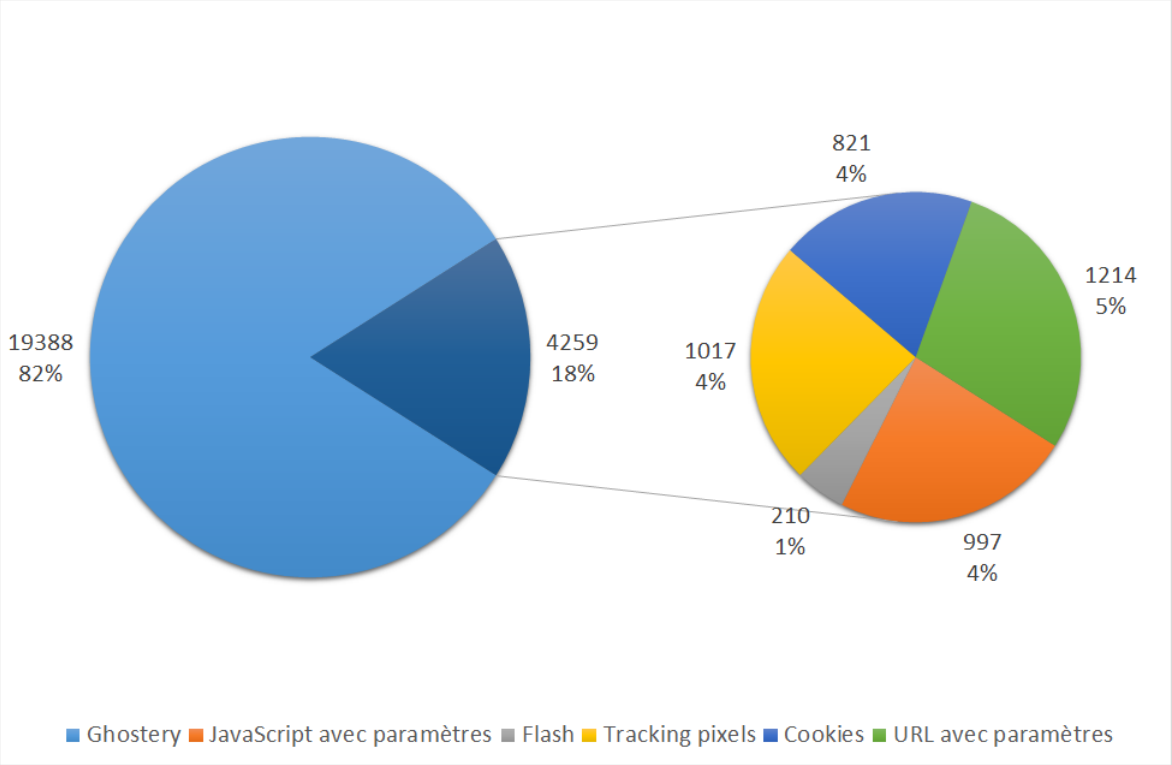

FIGURE 6.6 – Répartition des trackers par catégorie.

L'extension a laissé passer 19388 trackers connus de la base de données Ghostery et 4259 éléments ont été identifiés comme trackers selon les critères de l'outil.

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery	19388	81,99	- 21,27%
2	URL avec paramètres	1214	5,13	- 10,07%
3	Tracking pixels	1017	4,30	- 16,09%
4	JavaScript avec paramètres	997	4,22	- 7,60%
5	Cookies	821	3,47	- 7,55%
6	Flash	210	0,89	- 14,63%
	TOTAL	23647	-	- 19,57%

Rang	Types MIME (Ghostery)	#	%	Évolution
1	Images .gif	5817	30,00	- 21,98%
2	JavaScript	5659	29,19	- 18,68%
3	HTML	3782	19,51	- 21,70%
4	Type inconnu	1142	5,89	- 30,54%
5	Texte	826	4,26	- 22,73%

HTTPS Everywhere n'est pas une extension dont le but est de protéger la vie privée mais la motivation pour cette expérience était de voir si le fait de se connecter en HTTPS à des sites avait une incidence sur le nombre de trackers. D'après les résultats, il semble que oui car on observe une baisse de 19,57% du nombre de trackers par rapport à la référence.

6.1.6 Priv3

Présentation

Priv3 [8] est une extension uniquement disponible pour Firefox. Son but est de bloquer le tracking effectué par les réseaux sociaux. Priv3 ne bloque pas complètement toutes les interactions tierces, elle supprime de manière sélective l'inclusion de cookies tiers quand le navigateur récupère du contenu provenant des réseaux sociaux mais les réactive lorsqu'on veut interagir avec les modules des réseaux sociaux.

L'extension gère les réseaux sociaux suivants : Facebook, Twitter, Google+ et LinkedIn. Cependant, elle ne semble plus mise à jour depuis juillet 2011.

8. `http://priv3.icsi.berkeley.edu/`

Configuration de l'extension

Aucun paramètre de configuration n'est disponible pour cette extension.

Résultats - Figure 6.7

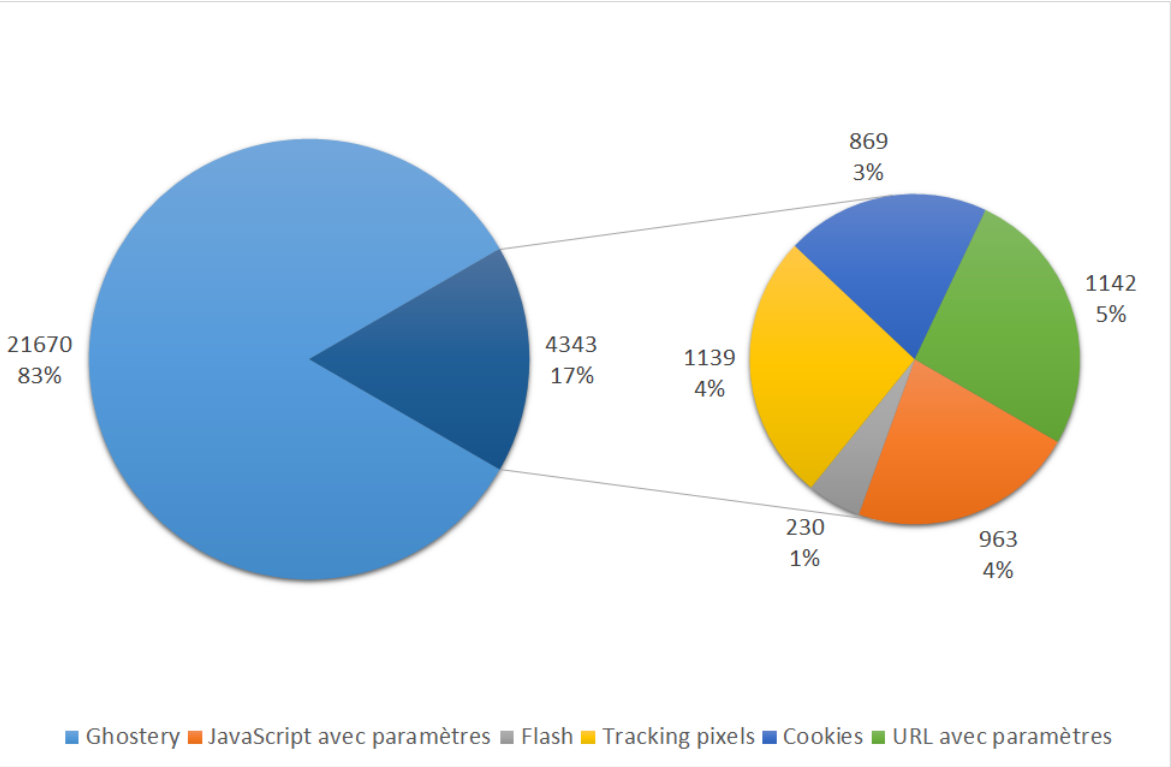

FIGURE 6.7 – Répartition des trackers par catégorie.

L'extension a laissé passer 21670 trackers connus de la base de données Ghostery et 4343 éléments ont été identifiés comme trackers selon les critères de l'outil.

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery	21670	83,30	- 12,01%
2	URL avec paramètres	1142	4,39	- 15,41%
3	Tracking pixels	1139	4,38	- 6,02%
4	JavaScript avec paramètres	963	3,70	- 10,75%
5	Cookies	869	3,34	- 2,14%
6	Flash	230	0,88	- 6,50%
	TOTAL	26013	-	- 11,53%

Rang	Types MIME (Ghostery)	#	%	Évolution
1	JavaScript	6484	29,92	- 6,83%
2	Images .gif	6137	28,32	- 17,69%
3	HTML	4307	19,88	- 10,83%
4	Type inconnu	1418	6,54	- 13,75%
5	Texte	954	4,40	- 10,76%

Priv3 a pour objectif de bloquer les trackers de 4 plateformes de réseaux sociaux. Le but de cette expérience n'était donc pas de voir si l'extension bloquait un nombre important de trackers mais plutôt d'estimer la proportion que représentent les réseaux sociaux dans les trackers. L'extension a bloqué 11,53% de trackers dont 12,01% de trackers connus dans la base de données Ghostery. Parmi ces derniers, les images .gif sont proportionnellement les plus bloquées avec une baisse de 17,69% mais ce sont les JavaScript qui sont en tête avec 6484 trackers de ce type ayant été détectés.

6.1.7 Privacy Badger

Présentation

Privacy Badger [9] est une extension disponible pour Firefox et Google Chrome, elle est actuellement en version alpha. Son but est de bloquer les trackers de domaines tiers grâce à l'analyse de leur comportement. Lors de l'envoi d'une requête, un entête Do Not Track est ajouté et l'extension évalue la probabilité d'être traqué grâce à des heuristiques. L'extension enregistre diverses informations afin d'affiner son fonctionnement en fonction des sites visités. Il faut donc un certain temps pour que l'utilisateur soit mieux protégé.

L'extension contient aussi une liste blanche pour certains domaines qui fournissent des ressources tierces essentielles. A long terme, les développeurs espèrent se séparer de cette liste grâce aux promesses des gestionnaires de sites de respecter Do Not Track. Le but de Privacy Badger n'est pas de bloquer les publicités mais de prévenir les invasions indésirables dans la vie privée des utilisateurs. Actuellement, seul le tracking provenant de tiers est bloqué. Dans le futur, les développeurs comptent également ajouter des protections pour les sites visités et ils vont implémenter des contre-mesures pour bloquer les fingerprinters dans une prochaine version.

9. `https://www.eff.org/privacybadger`

Configuration de l'extension

L'extension est restée configurée avec les paramètres par défaut.

Résultats - Figure 6.8

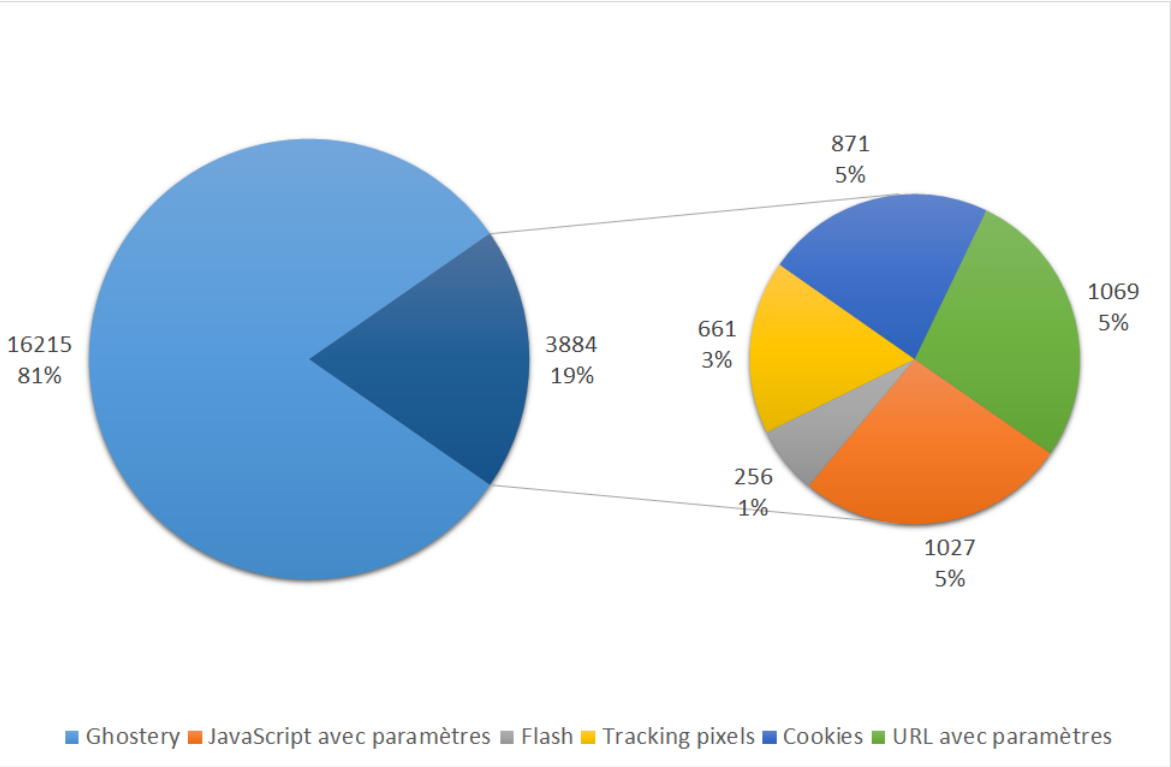

FIGURE 6.8 – Répartition des trackers par catégorie.

L'extension a laissé passer 16215 trackers connus de la base de données Ghostery et 3884 éléments ont été identifiés comme trackers selon les critères de l'outil.

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery	16215	80,68	- 34,16%
2	URL avec paramètres	1069	5,32	- 20,81%
3	JavaScript avec paramètres	1027	5,11	- 4,82%
4	Cookies	871	4,33	- 1,91%
5	Tracking pixels	661	3,29	- 45,46%
6	Flash	256	1,27	+ 4,07%
	TOTAL	20099	-	- 31,64%

Rang	Types MIME (Ghostery)	#	%	Évolution
1	JavaScript	5490	33,86	- 21,11%
2	Images .gif	4716	29,08	- 36,75%
3	HTML	2851	17,58	- 40,97%
4	Type inconnu	900	5,55	- 45,26%
5	Texte	601	3,71	- 43,78%

L'extension bloque un peu moins d'un tiers des trackers. Les trackers connus de Ghostery sont en tête des trackers détectés avec un nombre de 16215 mais ils ne représentent qu'une baisse de 34,16% alors que les pixels de traçage sont mieux bloqués avec une baisse de 45,46%. Le taux de détection de fichiers Flash est en hausse, cela s'explique par un nombre plus important de fichiers analysés (9 fichiers supplémentaires) par le *parser*. Les résultats de cette expérience sont faibles mais l'extension est encore en version alpha et il serait intéressant de refaire un test sur une prochaine version pour constater s'il y a une amélioration. Cette extension se basant sur le comportement des sites, elle mérite une attention particulière.

6.1.8 Adblock & Ghostery

Configuration des extensions

Adblock a été configuré avec le blocage des publicités jugées acceptables et la liste Fanboy Ultimate a été installée (de manière similaire à la troisième expérience d'Adblock). Ghostery a été configuré pour bloquer tous les trackers et tous les cookies (comme lors de l'expérience n'impliquant que Ghostery).

Résultats - Figure 6.9

Les extensions ont laissé passer 448 trackers connus de la base de données Ghostery et 1457 éléments ont été identifiés comme trackers selon les critères de l'outil.

Rang	Répartition des trackers	#	%	Évolution
1	JavaScript avec paramètres	509	26,72	- 52,83%
2	URL avec paramètres	471	24,72	- 65,11%
3	Ghostery	448	23,52	- 98,18%
4	Cookies	258	13,54	- 70,95%
5	Flash	123	6,46	- 50,00%
6	Tracking pixels	96	5,04	- 92,08%
	TOTAL	1905	-	- 93,52%

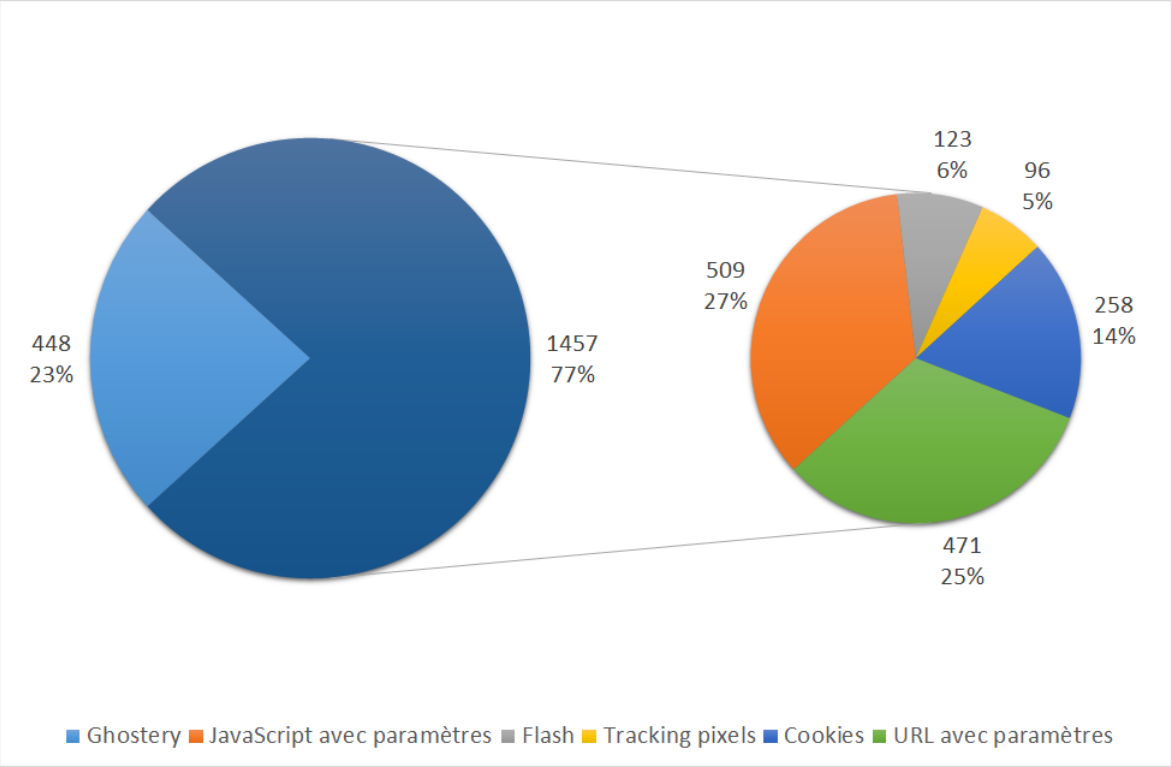

FIGURE 6.9 – Répartition des trackers par catégorie.

Rang	Types MIME (Ghostery)	#	%	Évolution
1	Images .jpeg	176	39,29	- 74,23%
2	Images .png	100	22,32	- 89,67%
3	JavaScript	63	14,06	- 99,09%
4	HTML	39	8,71	- 99,19%
5	Images .gif	31	6,92	- 99,58%

Cette expérience avait pour but de tester ensemble les deux extensions ayant les meilleurs résultats. Il en ressort que les extensions combinées apportent des résultats encore meilleurs car on atteint un taux de blocage de 93,52% (contre 87,63% pour Adblock et 92,44% pour Ghostery). Ceci s'explique par le fait qu'Adblock arrête plus de trackers qui ne sont pas détectés par Ghostery, notamment les pixels de traçage et les réponses HTTP qui créent un cookie tiers. D'un autre côté, Ghostery bloque des trackers qui ne sont pas présents dans les listes d'Adblock. Ces deux effets font que le nombre global de trackers diminue davantage afin d'atteindre le meilleur taux de protection.

6.2 Do Not Track

6.2.1 Présentation

Do Not Track [10] est un entête HTTP qui signale au serveur le souhait de l'utilisateur de ne pas être tracé. Les sites ont le choix d'accepter ce souhait ou de l'ignorer car il n'y a aucun effet contraignant s'ils ne le respectent pas.

L'activation se fait aisément dans les paramètres du navigateur, il ne s'agit que d'une simple case à cocher.

Le but de cette expérience était de voir si les sites respectent cet entête en regardant si le nombre de trackers détectés diminue de façon significative.

6.2.2 Résultats

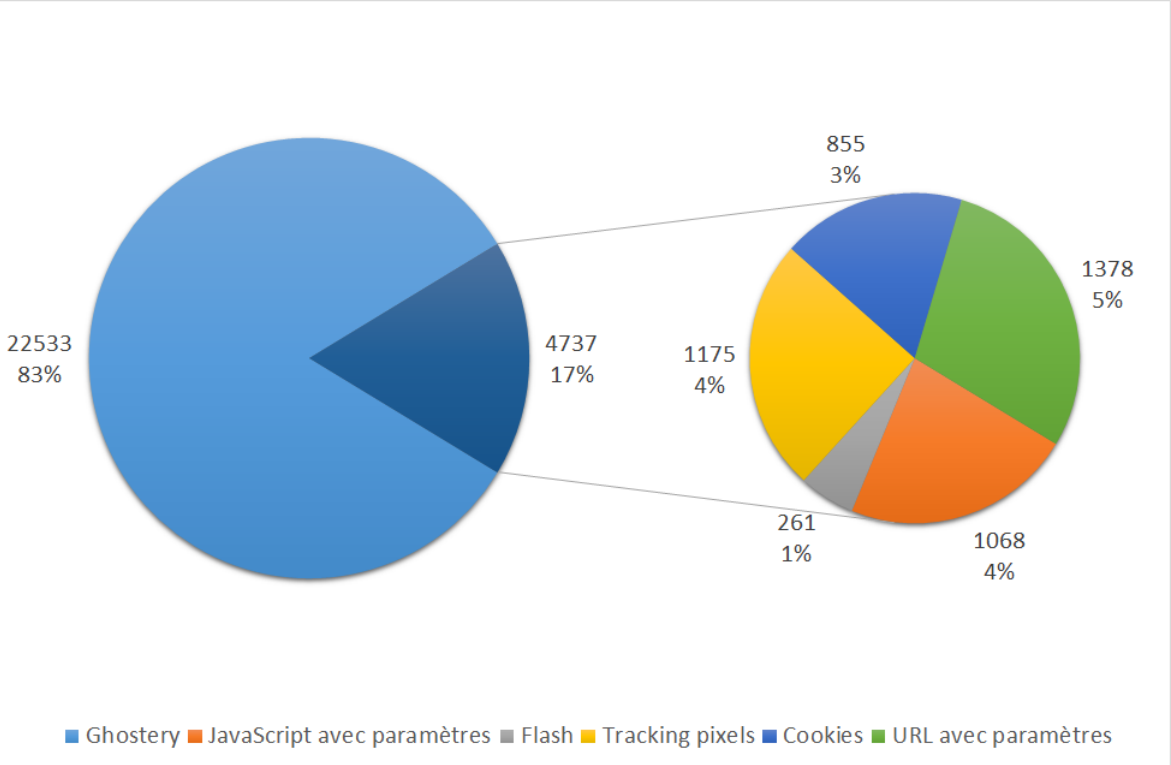

FIGURE 6.10 – Répartition des trackers par catégorie.

Do Not Track a laissé passer 22533 trackers connus de la base de données Ghostery et 4737 éléments ont été identifiés comme trackers selon les critères de l'outil.

10. `http://donottrack.us/`

Rang	Répartition des trackers	#	%	Évolution
1	Ghostery	22533	82,63	- 8,50%
2	URL avec paramètres	1378	5,05	+ 2,07%
3	Tracking pixels	1175	4,31	- 3,05%
4	JavaScript	1068	3,92	- 1,02%
5	Cookies	855	3,14	- 3,72%
6	Flash	261	0,96	+ 6,10%
	TOTAL	27270	-	- 7,25%

Rang	Types MIME (Ghostery)	#	%	Évolution
1	Images .gif	6597	29,28	- 11,52%
2	JavaScript	6584	29,22	- 5,39%
3	HTML	4473	19,85	- 7,39%
4	Type inconnu	1151	5,11	- 29,99%
5	Texte	1006	4,46	- 5,89%

Certains sites semblent respecter ce souhait car le nombre total de trackers détectés est en baisse de 7,25%. Comme pour les autres expériences, les pourcentages en hausse dans l'évolution par rapport aux chiffres de référence proviennent d'un nombre légèrement plus élevé de fichiers analysés.

Activer l'entête Do Not Track ne permet pas de se protéger de manière efficace contre les trackers mais il a l'avantage d'être très simple à activer. De plus, cela montre un signe encourageant de la part de certains sites qui commencent à prendre conscience que les utilisateurs désirent un meilleur respect de leur vie privée lorsqu'ils naviguent sur Internet.

6.3 Autres extensions intéressantes

D'autres extensions relatives à la protection de la vie privée sont intéressantes mais n'avaient pas de raisons d'être testées avec l'outil. Elles sont néanmoins présentées dans cette section à titre d'information.

6.3.1 BetterPrivacy

BetterPrivacy [11] est une extension qui a pour but de supprimer les cookies Flash après chaque session de navigation. La raison pour laquelle elle n'a pas été tes-

11. `https://addons.mozilla.org/fr/firefox/addon/betterprivacy/`

tée est que le *crawler* effectue l'ensemble des visites de sites au sein d'une même session. L'extension n'aurait donc pas pu être testée dans ses conditions optimales d'utilisation.

6.3.2 NoScript

NoScript [12] est une extension disponible pour Firefox qui a pour but de n'autoriser l'exécution que des scripts provenant de sites de confiance. Le test de cette extension n'aurait pas eu beaucoup de valeur car celle-ci aurait bloqué l'ensemble des scripts, ce qui n'aurait pas été représentatif d'une utilisation normale d'Internet.

6.4 Résultats

	Ghostery	JS avec paramètres	Flash	Tracking pixels	Cookies	URL avec param.	TOTAL
Normal (sans extension)	24627	1079	246	1212	888	1350	29402
Adblock (par défaut)	2584	558	118	89	303	563	4215
Adblock (pas de publicité acceptable)	2572	544	134	100	207	570	4127
Adblock (Fanboy Ultimate)	2197	497	139	99	168	538	3638
DoNotTrackMe	9341	812	227	424	567	852	12223
Ghostery	471	553	130	200	319	551	2224
HTTPS Everywhere	19388	997	210	1017	821	1214	23647
Priv3	21670	963	230	1139	869	1142	26013
Privacy Badger	16215	1027	256	661	871	1069	20099
Adblock & Ghostery	448	509	123	96	258	471	1905
DNT	22533	1068	261	1175	855	1378	27270

FIGURE 6.11 – Comparaison des différents expériences (avec la base Ghostery).

	JS avec paramètres	Flash	Tracking pixels	Cookies	URL avec param.	TOTAL
Normal (sans extension)	4966	670	7809	8690	4788	26923
Adblock (par défaut)	924	131	169	348	812	2384
Adblock (pas de publicité acceptable)	907	152	171	274	814	2318
Adblock (Fanboy Ultimate)	664	155	168	218	772	1977
DoNotTrackMe	2290	358	1951	3296	1525	9420
Ghostery	558	130	210	328	558	1784
HTTPS Everywhere	4016	485	5926	6406	3755	20588
Priv3	4472	574	6382	8076	3811	23315
Privacy Badger	3793	411	4478	5571	2717	16970
Adblock & Ghostery	512	123	101	260	474	1470
DNT	4701	688	6798	7072	4604	23863

FIGURE 6.12 – Comparaison des différents expériences (sans la base Ghostery).

Seule l'analyse utilisant la base de données Ghostery a été utilisée pour le calcul des résultats pour la raison évoquée dans la section 6.1. Cependant, ces deux tableaux permettent de constater que les deux types d'analyses fournissent proportionnellement les mêmes résultats sur les différentes expériences.

12. `http://noscript.net/`

Ils permettent également de voir en un seul coup d'œil quelles sont les expériences qui donnent les meilleurs résultats.

La combinaison des extensions Adblock (avec la liste Fanboy Ultimate) et Ghostery offre la meilleure protection.

Si l'utilisateur ne désire installer qu'une seule extension, son choix devrait se porter sur Ghostery. En effet, c'est l'extension qui obtient les meilleurs résultats même dans le type d'analyse qui n'utilise pas leur base de données. Elle est suivie par Adblock (équipée de la liste Fanboy Ultimate).

Chapitre 7

Conclusion

Il est incontestable que la vie privée sur Internet est un sujet d'actualité. Un nombre croissant d'utilisateurs émettent le souhait d'un meilleur respect de leur vie privée. Actuellement, on peut considérer que celle-ci est malmenée principalement pour des raisons financières. Des organisations n'hésitent pas à déployer des moyens afin de s'enrichir grâce à certaines informations précieuses.

Afin d'identifier les techniques utilisées pour effectuer ce tracking, il est nécessaire de comprendre les rudiments du fonctionnement du Web. Celui-ci a été amélioré avec le temps mais malgré cela, les mécanismes qui le composent ne sont parfois plus adaptés à l'environnement actuel. Nous avons vu les techniques principalement utilisées dans le traçage des utilisateurs. Étant nombreuses et variées, elles permettent d'apporter des informations différentes sur l'utilisateur et certaines sont considérées comme étant plus agressives que d'autres vis-à-vis du respect de la vie privée. Nous avons également constaté que certaines techniques sont moins utilisées avec le temps alors que d'autres sont en plein essor.

Dans le but de découvrir comment le traçage des utilisateurs est opéré sur le Web, un outil a été développé. Afin de déterminer si une ressource chargée joue le rôle de tracker, l'outil repose sur deux méthodes différentes : la première utilise une base de données de trackers tandis que l'autre repose sur des critères établis d'après les techniques généralement utilisées à des fins de tracking. Prises séparément, ces deux méthodes ne rapportent pas le même nombre de trackers mais chacune a des avantages et des inconvénients. Dans cet outil, les deux méthodes ont été unies afin d'offrir un taux de détection supérieur.

L'outil a ensuite été lancé sur le TOP 1000 des sites les plus visités au monde. Une analyse des résultats a notamment permis de montrer que l'outil était fiable et a mis au jour des erreurs contenues dans les réponses HTTP reçues des serveurs. Néanmoins, la meilleure chose que cette analyse a apporté est une image globale du niveau de traçage sur l'ensemble de ces sites. Cette première analyse a d'ailleurs servi de référence pour des expériences réalisées ultérieurement.

En plus de la cartographie fournie par les résultats de l'outil, celui-ci a permis de tester une série d'extensions destinées à protéger la vie privée des utilisateurs lorsqu'ils naviguent sur Internet. Ces expériences ont montré que certaines extensions apportent une meilleure protection que d'autres. Un classement a été établi dans le but de montrer une vue globale du niveau de protection que procurent ces extensions. Les expériences ont également permis de constater que les mentalités changent avec notamment une diminution non négligeable du nombre de trackers constatée avec l'activation de l'entête Do Not Track. Le non-respect de cet entête n'implique pas d'effets contraignants mais certains sites décident néanmoins de respecter le choix de leurs visiteurs.

L'outil développé pour ce mémoire fournit une bonne base pour l'analyse du niveau de traçage sur Internet mais il peut être amélioré. Le *crawler* montre un taux de réussite meilleur avec des timeout plus longs mais certains sites restent toujours en échec. Il serait intéressant de comprendre pour quelles raisons. Concernant le *parser*, les critères pourraient être raffinés notamment en procédant à l'analyse des scripts importés depuis un autre domaine. En effet, certains scripts importés de domaines tiers sont inoffensifs mais néanmoins considérés comme des trackers. Analyser leur comportement permettrait de réduire le taux de faux positifs.

Ce mémoire offre une bonne vue d'ensemble des techniques utilisées dans le traçage des utilisateurs mais les techniques envisageables sont si nombreuses qu'il n'était malheureusement pas possible de toutes les inclure ni de les expliquer en détail. Néanmoins, ce travail offre une base solide pour toute personne souhaitant s'intéresser au respect de la vie privée.

Annexe A

Options de l'outil implémenté

Lors du lancement du *crawler*, l'utilisateur peut spécifier plusieurs arguments :

— Requis : le mode (*crawler* ou *parser*).

— Requis : le répertoire des fichiers.
Pour le *crawler*, il s'agit du répertoire où les fichiers seront enregistrés.
Pour le *parser*, il s'agit du répertoire contenant les fichiers à analyser.

— Optionnel : l'activation du mode *debug* qui va donner davantage de détails en cas de problème.

— Optionnel : l'affichage de l'aide.

— Requis pour le *crawler* : le profil de Firefox à utiliser.

— Requis pour le *crawler* : le fichier contenant la liste des sites web à visiter.

— Requis pour le *crawler* : le début de l'intervalle des sites à visiter.

— Requis pour le *crawler* : la fin de l'intervalle des sites à visiter.

— Requis pour le *crawler* : le nombre de sites à visiter entre chaque redémarrage de Firefox.

— Optionnel pour le *crawler* : le nombre maximal de tentatives par site web (lors d'un échec dû à un *timeout*). Par défaut : 1 tentative.

— Optionnel pour le *crawler* : la durée du timeout lors du chargement d'un site. Par défaut : 30 secondes.

— Optionnel pour le *parser* : le fichier contenant la liste des trackers de Ghostery.

— Optionnel pour le *parser* : l'affichage dans le terminal de tous les trackers identifiés pour chaque fichier analysé.

Annexe B

Modifications dans l'extension NetExport

harBuilder.js

Lignes 200 à 211 commentées :

```
1  /*if (!timings._timeStamps)
2  {
3    timings.comment = "_timeStamps field contains timing data
       generated using " + "console.timeStamp() method. See
       Firebug documentation: " + "http://getfirebug.com/wiki/
       index.php/Console_API";
4    timings._timeStamps = [];
5  }
6
7  timings._timeStamps.push({
8    time: stamp.time - this.startedDateTime,
9    label: label
10 });*/
```

automation.js

Ligne 171 modifiée :

```
1  var fileName = name; /*+ "+" + now.getFullYear() + "-" + f(now
     .getMonth()+1) + "-" + f(now.getDate()) + "+" + f(now.
     getHours()) + "-" + f(now.getMinutes()) + "-" + f(now.
     getSeconds());*/
```

Annexe C

Format des résultats

Il existe deux types de résultats : les résultats détaillés pour chaque fichier HTTP Archive analysé et les résultats globaux de l'analyse.

Les résultats globaux sont enregistrés dans le dossier "logs" alors que les résultats détaillés sont enregistrés dans le dossier "results". Ceci a été décidé par souci de simplicité car les résultats globaux sont ainsi directement accessibles et ne sont pas noyés dans le dossier contenant les résultats détaillés.

Crawler

Les résultats globaux dans le dossier "logs" :

— **stats_flash-cookies.csv** contient la liste des sites utilisant des cookies Flash, triés par ordre décroissant.

Parser

Les résultats globaux dans le dossier "logs" :

— **stats_detailed.csv** contient la liste des sites avec le nombre détaillé d'éléments enregistrés (trackers connus identifiés avec l'aide de Ghostery, réponses HTTP créant un cookie, fichiers JavaScript avec ou sans paramètres chargés depuis un autre domaine, Flash chargés depuis un autre domaine, pixels espions et requêtes d'URL avec des paramètres).

— **stats_mimetypes_ghostery.csv** contient la liste des types d'éléments (*mimetype*) des trackers détectés par Ghostery, triés par ordre décroissant (si Ghostery a été utilisé par le *parser*).

— **stats_mimetypes_soa.csv** contient la liste des types d'éléments (*mimetype*) chargés d'un domaine différent, triés par ordre décroissant.

— **stats_trackers.csv** contient la liste des trackers identifiés grâce à Ghostery, triés par ordre décroissant (si Ghostery a été utilisé par le *parser*).

Les résultats détaillés pour chaque site dans le dossier "results" :

— **<URL du site>_cookies.csv** contient la liste des cookies créés (avec leurs détails) par des réponses HTTP d'un domaine différent.

— **<URL du site>_flash.csv** contient la liste des fichiers Flash provenant d'un autre domaine.

— **<URL du site>_ghostery.csv** contient la liste des trackers détectés grâce à la base de données Ghostery (si Ghostery a été utilisé par le *parser*).

— **<URL du site>_js.csv** contient la liste des fichiers JavaScript provenant d'un domaine tiers.

— **<URL du site>_js-query.csv** contient la liste des fichiers JavaScript provenant d'un domaine tiers et appelés avec des paramètres.

— **<URL du site>_mimetypes.csv** contient la liste des types d'éléments (*mimetype*) chargés d'un domaine différent, triés par ordre décroissant.

— **<URL du site>_parameters.csv** contient la liste des requêtes vers un domaine différent dont l'URL contient des paramètres.

— **<URL du site>_pixels.csv** contient la liste des pixels espions détectés depuis un autre domaine.

— **<URL du site>_urls.csv** contient la liste de l'URL de toutes les ressources chargées d'un domaine différent.

Annexe D

Trackers détectés par Ghostery pour les principaux types MIME

image/gif (Images .gif) :

http ://apx.moatads.com/pixel.gif ?e=17&i=AOL2&cm=1&bq=2&f=[...]

http ://ums.adtech.de/mapuser/providerid=1037 ;userid=16c29fb9-8d[...]

http ://pixel.rubiconproject.com/tap.php ?v=11581&nid=2395&put=[...]

text/javascript :

http ://beacon-5.newrelic.com/1/a2134792db ?a=163769&ap=479&be=1349&fe=2[...]

http ://data103.adlooxtracking.com/ads/check/check.php ?version=3&client=ebuzz[...]

http ://rma-api.gravity.com/v1/beacons/initialize ?u=undefined&sg=95ec266b244d[...]

application/x-javascript :

http ://at.atwola.com/addyn/3.0/5113.1/221794/0/-1/size=6x2 ;noperf=1 ;alias=93[...]

http ://tags.mathtag.com/notify/js ?exch=adt&id=5aW95q2jLzQvIC9ZVFl5Wmpk[...]

http ://assets.ebz.io/ebzFormats/buzzplayer/YoutubeBuzzPlayer.js ?2.0.aab1d6fd2d[...]

application/javascript :

http ://adadvisor.net/adscores/g.js ?sid=9201023828

http ://j.adlooxtracking.com/ads/js/tfa_ebuzz_creaebz.js

http ://tacoda.at.atwola.com/rtx/r.js ?cmd=ADG&si=1808&pu=http%3A//www.h[...]

text/html (HTML) :

https ://www.facebook.com/plugins/like.php ?action=like&app_id=46744042133&[...]

http ://ads.tw.adsonar.com/adserving/getAds.jsp ?previousPlacementIds=&placem[...]

http ://rma-api.gravity.com/v1/beacons/log ?cbust=955-45&site_guid=95ec266b24[...]

application/x-unknown-content-type (Type inconnu) :

http ://dtm.cb2.com/dmm/casale/match ?fpc=2437&pnid=19998&trid=476302305[...]

http ://at.atwola.com/addyn/3.0/5113.1/221794/0/-1/noperf=1 ;alias=93314234 ;cf[...]

http ://pixel.quantserve.com/pixel ;r=962460299 ;a=p-6fTutip1SMLM2 ;labels=[...]

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] Moxie Marlinspike. Conférence DEF CON 18 - Changing threats to privacy. https://www.youtube.com/watch?v=eG0KrT6pBPk.

[4] Moxie Marlinspike. Why 'I Have Nothing to Hide' Is the Wrong Way to Think About Surveillance. http://www.wired.com/2013/06/why-i-have-nothing-to-hide-is-the-wrong-way-to-think-about-surveillance/, June 2013.

[5] Moxie Marlinspike. Site personnel de Moxie Marlinspike. http://www.thoughtcrime.org/.

[6] Richard Esguerra. Google CEO Eric Schmidt Dismisses the Importance of Privacy. https://www.eff.org/deeplinks/2009/12/google-ceo-eric-schmidt-dismisses-privacy, December 2009.

[7] Eben Moglen. Snowden and the Future : Part II ; Oh, Freedom. http://snowdenandthefuture.info/PartII.html, October 2013.

[8] Alessandro Acquisti. Why privacy matters. http://www.ted.com/talks/alessandro_acquisti_why_privacy_matters, June 2013.

[9] Bruce Schneier. The Eternal Value of Privacy. https://www.schneier.com/essay-114.html, May 2006.

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

[11] 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.

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

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

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

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

[16] Google Analytics Cookie Usage on Websites - Google Analytics – Google Developers. `https://developers.google.com/analytics/devguides/collection/analyticsjs/cookie-usage`.

[17] 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.

[18] M. Zalewski. *The Tangled Web : A Guide to Securing Modern Web Applications*. No Starch Press, 2012.

[19] Same-origin policy sur le site de Mozilla. `https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy`.

[20] Shuo Chen, David Ross, and Yi-Min Wang. An Analysis of Browser Domain-isolation Bugs and a Light-weight Transparent Defense Mechanism. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 2–11, New York, NY, USA, 2007. ACM.

[21] B. Sullivan and V. Liu. *Web Application Security, A Beginner's Guide*. Beginner's Guide. McGraw-Hill Education, 2011.

[22] Chuan Yue, Mengjun Xie, and Haining Wang. Automatic Cookie Usage Setting with CookiePicker. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '07, pages 460–470, Washington, DC, USA, 2007. IEEE Computer Society.

[23] 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.

[24] William West and S. Monisha Pulimood. Analysis of privacy and security in html5 web storage. *J. Comput. Sci. Coll.*, 27(3) :80–87, January 2012.

[25] Mika Ayenson, Dietrich James Wambach, Ashkan Soltani, Nathan Good, and Chris Jay Hoofnagle. Flash Cookies and Privacy II : Now with HTML5 and ETag Respawning. July 2011.

[26] Dongseok Jang, Ranjit Jhala, Sorin Lerner, and Hovav Shacham. An Empirical Study of Privacy-violating Information Flows in JavaScript Web Applications. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 270–283, New York, NY, USA, 2010. ACM.

[27] Minh Tran, Xinshu Dong, Zhenkai Liang, and Xuxian Jiang. Tracking the trackers : Fast and scalable dynamic analysis of web content for privacy violations. In *Proceedings of the 10th International Conference on Applied Cryptography and Network Security*, ACNS'12, pages 418–435, Berlin, Heidelberg, 2012. Springer-Verlag.

[28] Facebook SDK pour JavaScript. `https://developers.facebook.com/docs/javascript`.

[29] API JavaScript de Google. `https://developers.google.com/+/web/api/javascript`.

[30] API JavaScript de LinkedIn. `http://developer.linkedin.com/javascript`.

[31] Introduction à Google Analytics. `https://developers.google.com/analytics/devguides/collection/analyticsjs/`.

[32] Adobe Flash Player. `http://get.adobe.com/fr/flashplayer/`.

[33] Lecteur vidéo HTML5 YouTube. `https://www.youtube.com/html5`.

[34] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash Cookies and Privacy. In *AAAI Spring Symposium : Intelligent Information Privacy Management*. AAAI, 2010.

[35] Adobe Flash Platform : Shared objects. `http://help.adobe.com/en_US/as3/dev/WS5b3ccc516d4fbf351e63e3d118a9b90204-7d80.html`.

[36] Google Chrome : Plug-in Adobe Flash Player. `https://support.google.com/chrome/answer/108086`.

[37] Adobe - Private browsing in Flash Player 10.1. `http://www.adobe.com/devnet/flashplayer/articles/privacy_mode_fp10_1.html`.

[38] Jonathan R. Mayer and John C. Mitchell. Third-party web tracking : Policy and technology. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 413–427, Washington, DC, USA, 2012. IEEE Computer Society.

[39] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster : Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 541–555, Washington, DC, USA, 2013. IEEE Computer Society.

[40] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. User tracking on the web via cross-browser fingerprinting. In *Proceedings of the 16th Nordic Conference on Information Security Technology for Applications*, NordSec'11, pages 31–46, Berlin, Heidelberg, 2012. Springer-Verlag.

[41] Peter Eckersley. How unique is your web browser ? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 1–18, Berlin, Heidelberg, 2010. Springer-Verlag.

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

[43] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective : Dusting the Web for Fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 1129–1140, New York, NY, USA, 2013. ACM.

[44] Selenium - Web Browser Automation. `http://docs.seleniumhq.org/`.

[45] Spécification du format HTTP Archive. `http://www.softwareishard.com/blog/har-12-spec/`.

[46] Firebug : extension Firefox. `https://getfirebug.com/`.

[47] Firebug extensions : NetExport. `https://getfirebug.com/wiki/index.php/Firebug_Extensions#NetExport`.