Sécurité des applications AJAX

Unity Web Sàrl

Travail de diplôme réalisé en vue de l'obtention du diplôme HES

par

Cem KOKER

Conseiller au travail de diplôme :

M. Rolf HAURI

Genève, le 19 décembre 2007

Haute École de Gestion de Genève (HEG-GE)

Filière informatique de gestion

Déclaration

Ce travail de diplôme est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du diplôme HES. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de diplôme, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de diplôme, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 19 décembre 2007

Cem KOKER

_____

Remerciements

Un grand merci à toute l'équipe d'Unity Web Sàrl pour leurs soutiens techniques et leurs encouragements.

Remerciements particuliers à M. Hauri qui m'a permis de réaliser ce travail avec une grande autonomie et liberté et pour sa disponibilité lorsque cela a été nécessaire.

Sommaire

AJAX est un ensemble de technologies Internet déjà existantes depuis quelques années et remises au goût du jour il y a peu. Elles permettent de créer des flux de communication entre client et serveur de manière asynchrone. Grâce à AJAX, le client qui visite une page Internet qui utilise ces technologies n'a plus besoin d'attendre la réponse du serveur pour envoyer de nouvelles données.

Ce changement est radical dans le monde des communications entre client et serveur Web car les échangent étaient jusque là synchrones. La re-découverte de cette technologie, ou ensemble de technologies, a suscité un énorme intérêt de la communauté Internet pour son utilisation. Des géants d'Internet tels que Google ou Yahoo en ont même fait leur cheval de guerre et en ont démocratisé l'utilisation que nous en faisons aujourd'hui.

Ce travail de diplôme propose de faire un bilan des quelques mois qui ont suivi la re-découverte de ces technologies et leurs utilisations massive sur Internet. De sa découverte à son évolution future en observant les alternatives; il sera question de parler des avantages et des inconvénients que son utilisation apporte du point de vue de l'utilisateur aussi bien que celui du concepteur de logiciel Web.

La sécurité sera un point majeur de ce travail car elle constitue aujourd'hui un des enjeux majeurs de l'utilisation d'AJAX. Son adoption parfois hâtive et non réfléchie pour suivre l'effet de mode n'a pas été sans conséquences, elles seront décrites dans ce travail qui en dressera une liste non exhaustive quant aux problèmes qui peuvent se poser. Ils seront illustrés par des exemples frappants qui ont fait trembler plusieurs grands acteurs d'Internet. Ces derniers serviront d'introduction à une étude plus détaillée des différentes failles de sécurité qu'AJAX aura pu créer et d'en expliquer les tenants et les aboutissants en expliquant quelles sont les faiblesses de leurs implantations tout en décrivant leurs schémas d'attaques.

Suivra une réflexion sur les moyens à mettre en œuvre afin de se prémunir contre l'exploitation de ces failles en expliquant concrètement comment bien implanter son utilisation.

Enfin, dans le but d'aider le concepteur, nous réfléchirons à la possibilité d'automatiser la recherche de failles de sécurité dans la conception d'applications AJAX en définissant quelques lignes directrices à respecter lors de son implantation. Le but est

de mettre en avant les problèmes liés à la sécurité tels que des failles que le code pourrait comporter et qui seraient facile à détecter. Il serait aussi possible de réfléchir à la création d'une extension au navigateur Internet destiné à montrer à l'utilisateur les diverses interactions qui se déroulent à son insu.

Table des matières

Déclaration i
Remerciements ii
Sommaire iii
Table des matières 1
Liste des Figures 3
1. Internet 2.0 4
2. Qu'est-ce qu'AJAX ? 6
2.1 La redécouverte 6
2.2 Technologie 9
2.3 Avantages et inconvénients 11
2.4 Quelques applications qui utilisent la technologie AJAX 12
2.5 L'effet de mode 13
2.6 Les alternatives 13
2.6.1 Alternatives d'Adobe 13
2.6.2 Alternatives de Microsoft 14
2.7 L'évolution 15
3. Problématique 16
3.1 Surface d'attaque élargie 16
3.2 Les Sessions 17
3.2.1 Piratage de sessions 17
3.2.2 XSS ou « Cross Site Scripting » 18
3.3 Déni de service 19
3.4 Deux géants se font avoir 20
3.4.1 Yamanner, virus, Yahoo mail 20
3.4.2 Samy, ce héro ! 21
3.5 Vie privée 21
3.5.1 Espionnage à l'insu de l'utilisateur 21
4. Solutions 23
4.1 Protection de la vie privée 23
4.2 Gestion des sessions 23
4.3 Une bonne Implémentation 24
4.3.1 Schémas d'authentification 24
4.3.2 Ne jamais faire confiance au client 24
4.3.3 Logique business sur le serveur 25
4.3.4 Légitimité des requêtes 25
4.3.5 Validation des données 25
4.4 Du côté des navigateurs 26
4.5 Aide à la conception 26

4.5.1 Utilisation d'un framework existant *26*
4.5.1.1 jQuery 26
4.5.1.2 ExtJS 27
4.5.1.3 Autres 28
4.5.2 Frameworks – gage de sécurité *28*
5. Démonstration d'une attaque **29**
6. Aide à la sécurisation d'AJAX **30**
6.1 Outil d'analyse syntaxique **30**
6.2 Proxy applicatif **30**
6.3 Extension navigateur **31**
7. Conclusion **32**
8. Bibliographie **34**
Annexe 1 Les attaques contre le protocole SSL **35**

Liste des Figures

Figure 1 Web 1.0 et 2.0 5

Figure 2 Exemple de données XML 7

Figure 3 Exemple de données JSON 8

Figure 4 Fonctionnement d'AJAX 9

Figure 5 Architecture d'Ajax 10

Figure 6 Google Maps 12

Figure 7 Exemple d'application Flex 14

Figure 8 Déni de service 19

Figure 9 jQuery Framework 27

Figure 10 ExtJS Framework 28

1. Internet 2.0

Internet a longtemps été perçu par les internautes comme un ensemble de pages créés par des Webmasters et peu mises à jour. Internet était jusqu'à peu un espace très individualiste où la communication entre Webmasters et visiteurs était souvent à sens unique. Le nombre de personnes connectées n'a cessé de croître.

> *« … Quelque 694 millions de personnes dans le monde, âgées de plus de 15 ans, … , utilisent Internet, soit 14 % de la population mondiale de cette tranche d'âge, selon une étude publiée hier par ComScore Networks. … »*
>
> Source : ledevoir.com, 5 mai 2006

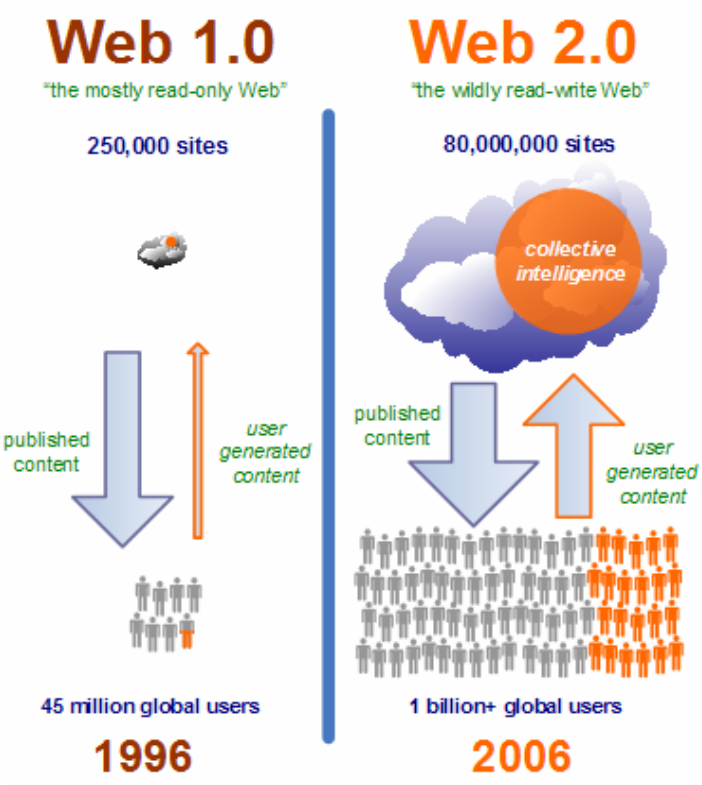
Ainsi en est également le nombre de sites créés. L'utilisation d'AJAX dans ces derniers a profondément modifié le visage d'Internet. On parle maintenant de Web communautaire et participatif. Il existait certes autrefois quelques communautés éparses mais aucune de l'ampleur qu'ont connus des sites comme Facebook ou MySpace qui comptent à eux seuls plus de 150 millions de membres.

On parle maintenant de Web communautaire où chacun peut librement s'exprimer et participer ainsi à la constitution d'une forme d'intelligence collective.

La *figure 1* illustre bien cette évolution, on remarque que le contenu est maintenant presque autant généré par les utilisateurs que les Webmasters.

Figure 1

Web 1.0 et 2.0



Source : http://Webilus.com/illustration/lintelligence-collective-du-Web20

Une très bonne illustration de ce qu'est le Web participatif est sans doute l'encyclopédie en ligne Wikipedia[1]. Sur Wikipedia il est en effet possible à tout un chacun de créer ou de compléter un article sur un sujet donné, ainsi ce sont les utilisateurs qui créent le contenu de l'encyclopédie qui est donc en perpétuelle évolution. Il est auto géré et des modérateurs valident ou invalident les articles, ces derniers sont élus au sens d'un vote démocratique sur le site.

Avec le nouveau Web, est apparue également une nouvelle terminologie, comme des « flux RSS » ou des « tag clouds[2] ». Les flux RSS permettent aux visiteurs de s'abonner à un site à la manière d'un journal hebdomadaire et de recevoir automatiquement les dernières publications de ce dernier. Les « tag clouds » quant à

[1] http://www.wikipedia.org – première encyclopédie participative au monde.

[2] Nuage de mot-clé – rassemblement de mots-clé

eux permettent d'organiser les pages au sein d'un site ou d'un moteur de recherche afin de les classifier selon des mots-clefs qui en facilitent la recherche.

Et enfin, au cœur de cette révolution : AJAX qui a permit de changer radicalement la face du Web. La suite de ce travail explique AJAX dans les moindres détails

2. Qu'est-ce qu'AJAX ?

2.1 La redécouverte

AJAX est l'acronyme de « **A**synchonous **J**avaScript Technology **A**nd **X**ML ». Il a été « formalisé » pour la première fois en février 2005 par la société américaine Adaptive Path [3] qui en expliqua les premiers fondements. Il s'agit d'une technologie ou plutôt d'un ensemble de technologies dont la re-découverte a complètement bouleversé la manière de travailler sur Internet. AJAX se base essentiellement sur deux technologies.

- Javascript, langage de programmation principalement utilisé sur les pages Internet, créé par Brendan Eich en 1995 basé sur Java mais simplifié pour en faciliter l'apprentissage pour les débutants. Une application écrite en Javascript est exclusivement destinée à s'exécuter sur le navigateur du client (ou visiteur de la page Internet).

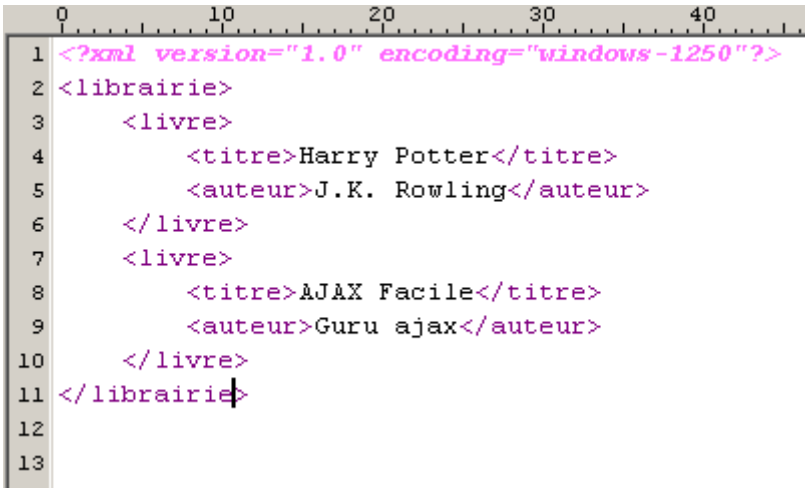
- Le langage XML qui est l'acronyme de e**X**tensible **M**arkup **L**anguage. XML permet le balisage universel de n'importe quelle Information (au sens large). XML a été créé par le World Wide Web Consortium dans le but de faciliter l'échange d'information sur Internet, plus précisément entre plusieurs systèmes d'information hétérogènes.

[3] http://www.adaptivepath.com

Un exemple de données XML illustré sur la *Figure 2*

Figure 2

Exemple de données XML



```
1  <?xml version="1.0" encoding="windows-1250"?>
2  <librairie>
3      <livre>
4          <titre>Harry Potter</titre>
5          <auteur>J.K. Rowling</auteur>
6      </livre>
7      <livre>
8          <titre>AJAX Facile</titre>
9          <auteur>Guru ajax</auteur>
10     </livre>
11 </librairie>
12
13
```

Les applications AJAX utilisent le plus souvent XML pour communiquer entre client et serveur. Toutefois, sont utilisation requiert l'interprétation et le décodage chez ces derniers qui n'est pas incluse par défaut et requiert des modules supplémentaires.

Une alternative à XML est JSON, acronyme de **J**ava**S**cript **O**bject **N**otation. Il s'agit d'un autre format de données générique. Léger et au format texte (facilement transmissible) il est facilement lisible et permets un apprentissage rapide.

En Javascript, il est de plus très facile d'utiliser des expressions JSON (que le client aurait reçus en répons du serveur par exemple). Une simple commande suffit :

```
var mesDonnees = eval('(' + ''donnees_json_recues'' + ')');
```

Cette évaluation comporte toutefois quelques risques :

> *« Cette méthode comporte toutefois des risques car la chaîne de caractères JSON peut contenir n'importe quel code JavaScript. Il existe une méthode plus sûre qui consiste à parser la chaîne de caractères donnees_json, seule solution disponible dans les autres langages de programmation à l'exception de Python, la syntaxe de JSON correspondant à ses deux types principaux : les listes et les dictionnaires. »*

Source : http://fr.wikipedia.org/wiki/JSON

Exemple de données JSON illustré sur la *Figure 3*

Figure 3

Exemple de données JSON

Format JSON :

```
{"menu": {
    "id": "file",
    "value": "File",
    "popup": {
      "menuitem": [
        {"value": "New", "onclick": "CreateNewDoc()"},
        {"value": "Open", "onclick": "OpenDoc()"},
        {"value": "Close", "onclick": "CloseDoc()"}
      ]
    }
 }}
```

À titre de comparaison, même exemple en XML :

```
<menu id="file" value="File">
    <popup>
      <menuitem value="New" onclick="CreateNewDoc()" />
      <menuitem value="Open" onclick="OpenDoc()" />
      <menuitem value="Close" onclick="CloseDoc()" />
    </popup>
 </menu>
```

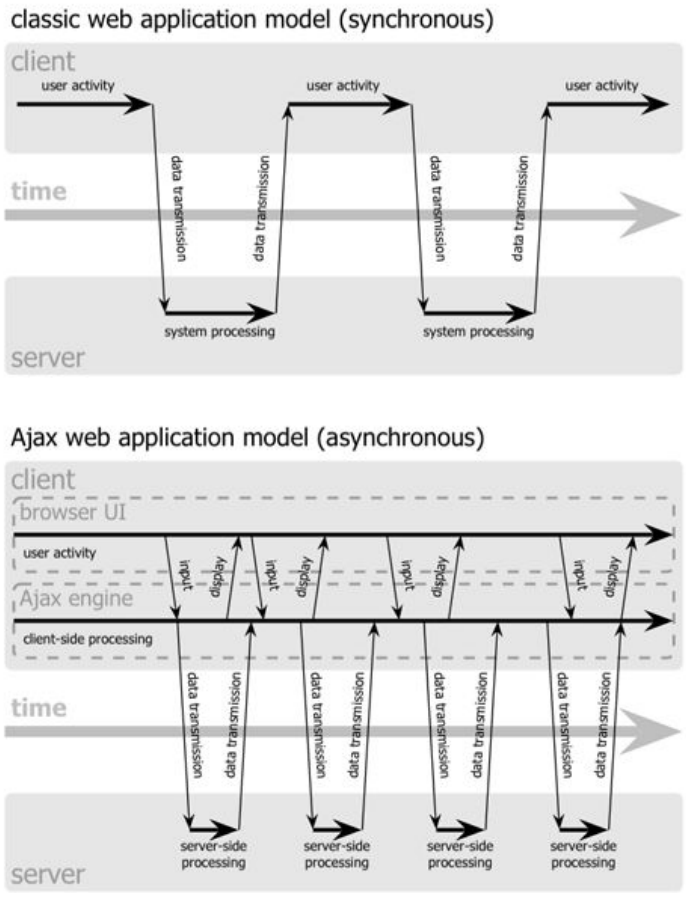
Source : http://fr.wikipedia.org/wiki/JSON

2.2 Technologie

Les échanges sur Internet avant l'utilisation d'AJAX fonctionnaient selon un mode de communication synchrone. Lorsque le client souhaitait accéder à une page Internet, il formulait une requête HTTP[4] à un serveur distant. Ce dernier effectuait ses traitements puis renvoyait une page au format HTML[5] au client ou plus précisément à son navigateur Internet qui se chargeait de l'afficher. On peut décrire ce mécanisme comme un système de « marche arrêt marche arrêt » dit « classique » illustré ici sur la *figure 4*

Figure 4

Fonctionnement d'AJAX



Source : http://www.adaptivepath.com/images/publications/essays/ajax-fig2.png

[4] Hypertext Transfer Protocol développé pour le World Wide Web

[5] Hypertext Markup Language, abgrégé HTML, est un langage informatique de balisage conçu pour écrire les pages Web

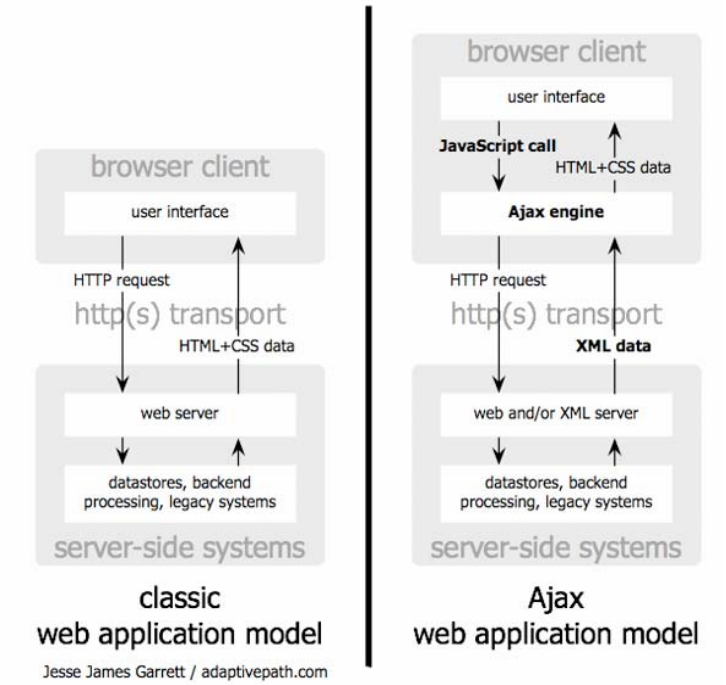
Cette approche est des plus frustrantes du point de vue l'interactivité entre l'utilisateur et l'application Web car une bonne partie de son temps est consacré à l'attente de la réponse du serveur. Jusqu'à l'apparition d'AJAX, personne ne s'était vraiment posé la question de savoir pourquoi les applications Web nécessitaient de recharger complètement une page à chaque interaction de l'utilisateur plutôt que de charger la totalité de l'application une fois pour toute et de simplement permettre à l'application Web d'interagir avec le serveur seulement lorsque ceci est nécessaire.

L'utilisation d'AJAX permet de supprimer ces temps d'attentes et de masquer aux yeux de l'utilisateur les communications avec le serveur. Le client qui croit que l'application s'exécute sur sa propre machine et n'a aucune idée des informations échangées avec le serveur. Cela pose un problème de respect de la vie privée puisque le client n'a aucune idée de la nature des données qu'il envoie. Ce problème fait l'objet d'un chapitre à part entière.

Techniquement parlant, l'application Web exécute un programme écrit en Javascript qui s'exécute chez le client. Ce dernier créer un objet AJAX : l'objet XMLHTTPRequest de Javascript puis utilise ce dernier pour communiquer avec le serveur. Cette 'couche' de communication entre le client et le serveur permet de gérer toutes les interactions entre le client et le serveur (voir Figure 5)

Figure 5

Architecture d'AJAX



Source : http://www.adaptivepath.com/images/publications/essays/ajax-fig1.png

Le fait de faire exécuter du code qui 'appartient' à l'application sur le navigateur du client permet également à ce dernier de pouvoir en voir le code source et d'en comprendre le fonctionnement. L'application franchit donc la barrière du serveur qui n'est plus en mesure de garantir l'authenticité du code qui est exécuté chez le client. Un client malveillant pourrait très bien disséquer la partie qu'il exécute et découvrir les points d'accès à l'application sur le serveur Web et de chercher à en exploiter les failles. Ces dernières font l'objet d'un chapitre séparé de ce travail.

2.3 Avantages et inconvénients

Quand on pense aux avantages qu'AJAX a pu apporter à l'utilisation d'Internet on pense tout de suite au gain phénoménal que ce dernier a pu apporter en matière d'interactivité. Avant AJAX, client et serveur devaient attendre à tour de rôle la fin du traitement de l'information de l'autre ce qui empêchait de faire autre chose durant ce temps. AJAX permets de traiter des flux d'informations de manière asynchrone et de manière parfaitement invisible à l'utilisateur qui a l'impression d'être en face d'une application comme si elle était installée sur son ordinateur.

Les inconvénients sont multiples. Une connexion permanente à Internet durant toute la durée des transactions est obligatoire car l'application a besoin de communiquer avec le client à tout instant. Toute interruption momentanée de la connexion peut entraîner des réactions imprévisibles de l'application[6].

La multiplication des requêtes a aussi pour effet pervers d'augmenter l'utilisation de la bande passante ce qui, pour des connexions à bas débits, peut être problématique.

Notons également que le client n'a pas aucune idée des données qui sont échangées avec le serveur. Il se peut que des applications permettent d'échanger des données auxquelles il n'aurait pas eu droit sans l'approbation de l'utilisateur. Ce problème est la cause principale des premières attaques basées sur AJAX et est encore à ce jour un problème qui n'a pas vraiment été prit en considération.

La sphère privée peut ne pas être respectée lors de l'utilisation d'une application AJAX. Ce problème sera analysé au cour de ce travail.

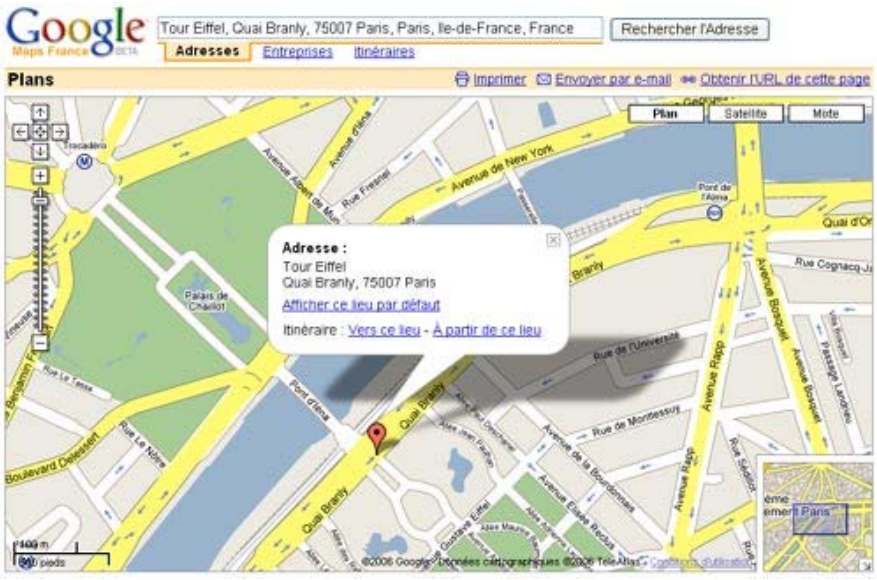
[6] A ce sujet, Google et d'autres sociétés sont en train de mettre en place des outils de connexion hors ligne qui permettrons de ne plus avoir besoin d'une connexion permanente. Plus d'informations sur : GoogleGears

2.4 Exemple d'utilisation de la technologie AJAX

L'une des premières applications les plus « impressionnantes » a sans doute été Google Maps (de Google).

Figure 6

Google Maps



Source : http://www.maps.google.fr

Cette application permet à l'utilisateur de chercher une adresse n' importe où sur la planète et de l'afficher sur une carte. On peut penser à juste titre que ce genre d'application n'a rien de bien révolutionnaire puisque déjà largement utilisée par d'autres sites comme par exemple le site de *Via Michelin*[7] ou le *plan de la ville de Genève*[8]. La vraie prouesse technologique a été de faire en sorte que le navigateur n'ait plus besoin de recharger complètement la page Web à chaque déplacement sur la carte[9]. En pratique, l'expérience en terme d'ergonomie qu'a apporté AJAX a permis de pouvoir se libérer des temps d'attente qui pouvaient subsister entre le 'clique' de l'utilisateur pour se déplacer dans la carte et son affichage effectif. Dorénavant l'utilisateur peut se déplacer librement sur la carte sans avoir besoin d'attendre son rechargement complet. La technique qui réside derrière est des plus simplistes. Il suffit

[7] http://www.viamichelin.com/

[8] http://w3public.ville-ge.ch/ville-ge/adrge.nsf

[9] Une idée pourtant simple, qui a valu à l'inventeur du DHTML d'être débauché de Microsoft pour travailler pour Google

de mettre en mémoire tampon toutes les possibilités de déplacement de l'utilisateur, permettant ainsi de ne pas avoir à charger les données à chaque déplacement sur la carte. AJAX a également permit de faciliter la validation de données, notamment lors du remplissage de formulaires. L'utilisateur peut dorénavant bénéficier d'une aide interactive lorsqu'il remplit les champs ce qui peut faire gagner du temps et peut éviter des erreurs de saisie. La contrepartie du gain d'interactivité est que l'application peut espionner l'utilisateur à son insu. Cette intrusion dans la vie privée fait l'objet d'un chapitre dédié dans ce travail.

2.5 L'effet de mode

Comme toute nouveauté, AJAX et son potentiel en terme d'ergonomie qu'il pouvait apporter sont devenus un « plus » que tout site Internet digne de ce nom devait avoir pour ne pas perdre risquer de perdre sa réputation vis-à-vis de ses visiteurs

C'est ainsi qu'en quelques mois une série de sites utilisant AJAX est apparue. Il s'agit sans doute de la plus grande révolution qu'Internet ait connue. Le nombre de sites utilisant cette technologie a cru de manière exponentielle et on dit qu'il se créer un blog toute les secondes.

2.6 Les alternatives

Il existe des alternatives, le plus souvent commerciales, à l'utilisation d'AJAX. Voici quelques-unes d'entres-elles :

2.6.1 Alternatives d'Adobe

Créée par Macromedia en 2004 (puis reprise par Adobe en 2006), Flex est une solution de développement qui permet de créer et de déployer des applications Internet riches (RIA). Elle repose essentiellement sur deux technologies :

- *MXML (qui est basé sur XML), qui permet la réalisation d'interfaces utilisateur très interactives. Son approche déclarative permet une très grande flexibilité et permet de contrôler tous les aspects de l'application. MXML est comparable à XUL de la Foundation Mozilla ou XAML de la société Microsoft.ActionScript 3.0 est un langage orienté objet similaire au Javascript et permettant de créer très rapidement des applications Internet riches.*

- *Pour sa partie présentation, Flex utilise la technologie Flash (97% des machines sont équipées du Flash Player), ce qui rend par conséquent les applications Flex relativement multi plates-formes et facilement déployables. Ce dernier point étant discutable, étant donnée l'installation nécessaire d'un plug-in Flash Player sur les machines clientes, ce qui*

n'est pas le cas avec une interface écrite selon la méthode AJAX car tous les navigateurs Web autorisent par défaut le JavaScript. Cependant, l'interprétation de Javascript étant différente selon les navigateurs, un des intérêts de Flex est de proposer une interface homogène, quel que soit l'équipement de l'utilisateur.

Source : http://fr.wikipedia.org/wiki/Adobe_Flex

Un exemple d'application Flex est visible sur la *Figure 7*.

Figure 7

Exemple d'application Flex

Sourrce : http://dashboardcompany.com/images/gallery/BW_Large_600x441.png

2.6.2 Alternatives de Microsoft

ActiveX aussi connu sous le nom de COM (Component Object Model) a été créé par Microsoft. Il permet le dialogue entre programmes. Implanté sur de nombreuses plates-formes il est utilisé la plupart du temps sous Windows.

ActiveX s'intègre à Internet Explorer en offrant de pouvoir ouvrir par exemple des documents Word à l'intérieur du navigateur Internet sans avoir à lancer la suite Microsoft Office. ActiveX permet également de créer des scripts qui peuvent gérer à leur tour des programmes écrits en Java ou commander des contrôles ActiveX.

ActiveX permet tout comme AJAX de d'envoyer des requêtes asynchrones à un serveur Web.

ActiveX n'est pas multi plateforme puisqu'il requiert d'avoir Internet Explorer et une plateforme Microsoft ce handicap profite à Javascript qui peut être exécuté indépendamment de la plateforme.

2.7 L'évolution

Le futur semble être des plus prometteur pour AJAX. On assiste un engouement très important de tous les acteurs d'Internet pour son développement. Il est fascinant d'observer à quel point l'architecture des applications a changée au fil des années.

Au début des années 80 l'architecture client serveur se résumait à un très gros mainframe[10] et des clients, appelés « terminaux », qui se connectaient simplement sur le mainframe, toute la puissance de calcul résidant sur le serveur principal. Puis, vers la fin des années 90, on a assisté à un déplacement de la puissance de calcul vers les clients. Ces derniers étaient (et le sont encore aujourd'hui) capables de traiter eux-mêmes les opérations qui nécessitaient autrefois un serveur avec une grande puissance de calcul.

De nos jours, avec l'arrivée d'AJAX et autres technologies de « remoting[11]», on assiste à un retour en arrière. Dans quelques années il ne sera peut-être plus nécessaire de disposer d'un ordinateur puissant pour pouvoir utiliser ses applications favorites. A ce sujet, Google (et autres concurrents) proposent déjà des solutions de bureautique entièrement « en ligne ». L'utilisateur n'a ni besoin d'installer sa suite bureautique usuelle (tableur, éditeur de texte, présentations) ni besoin de voyager avec ses documents en permanence. Tout est stocké sur le serveur Web qui rends donc les documents accessibles depuis n'importe quel navigateur Internet. Le fait d'avoir ses documents à la merci des éventuels pirates sans savoir « où » exactement ils sont stockés peut poser un réel problème quant à la sécurité et la sûreté des données.

Enfin on peut se poser la question de savoir si cette externalisation des données soit une bonne chose. On peut comprendre que certaines entreprises ne soient pas enclin à vouloir externaliser leurs données, or il s'agit bien là du problème car la plupart des applications Web qui utilisent Ajax sont louées au client comme un service.

Le site Blinksale[12] propose par exemple un service de gestion des factures en ligne. Il propose à ses clients de traiter et d'envoyer les factures directement depuis le site. Il connaît un succès colossal. On peut donc se demander si la sécurité des données préoccupe vraiment leurs clients.

---

[10] Ordinateur très puissant capable de traiter des millions d'informations par seconde

[11] … d'accès à distance des données …

[12] http://www.blinksale.com/home

3. Problématique

AJAX a certes permis d'apporter une nouvelle façon de concevoir et d'utiliser les applications Web, mais a également fait renaître des vieux démons du passé dont l'industrie se serait bien passée.

Avant l'apparition d'AJAX, l'utilisation de Javascript sur les pages Internet était souvent limité à la validation de formulaires HTML et autres menues actions souvent accessoires par rapport à la fonctionnalité principale de l'application Web.

Le fait d'utiliser AJAX dans les applications Web remet Javascript en tête de liste des langages utilisés du fait qu'AJAX se base sur ce dernier pour exécuter des programmes directement sur le navigateur du client. De ce fait, le concepteur n'a aucun moyen de vérifier l'authenticité du code que le client exécute car il ne peut pas garantir qu'il s'agit bien du code original qu'il a téléchargé. On peut comparer cela à un logiciel qui serait signé numériquement[13] et dont toute modification changerait la signature. Il est donc tout à fait possible que le client n'exécute pas l'application originale mais une version piratée, ce qui peut poser des problèmes en matière de sécurité.

3.1 Surface d'attaque élargie

On parle de surface d'attaque pour décrire l'ensemble des points d'entrées (failles ou portes dérobées) que le pirate pourrait utiliser afin de s'introduire dans un système informatique. Avec une application classique les flux d'entrée/sortie sont clairement définis et l'architecture de l'application reste simple avec un seul point d'entrée (synchrone)

Avec AJAX, l'application est devenue bien plus complexe, elle permet d'exécuter plusieurs commandes en simultané. L'attaquant peut donc désormais exploiter plusieurs points d'entrées. On peut comparer les applications dites classiques et les applications AJAX comme on comparerait un bunker avec une seule porte d'entrée, peu conviviale, et une maison moderne avec beaucoup de fenêtres. Il suffit ensuite d'imaginer que les fenêtres sont des nouvelles entrées possibles pour les voleurs, cette image permet de mieux comprendre que la complexité des applications AJAX rends la tâche de sécurisation de plus en plus complexe.

[13] Une signature numérique d'une application est sous la forme d'une combinaison de chiffres et de lettres qui garantissent que l'application est « originale », en effet, tout changement dans l'application modifie la signature qui est alors différente de l'originale

On pourra néanmoins énumérer quelques règles de base qu'il conviendra de suivre afin de ne pas permettre au pirate des actions malveillante elles seront décrites plus tard dans ce travail.

3.2 Les Sessions

Avec une application dite « classique » sans AJAX, le problème de la synchronisation des sessions ne se posait pas vraiment. Le serveur était en charge de gérer plusieurs sessions simultanées pour un utilisateur donné. Avec AJAX, il s'agit de gérer plusieurs sessions au sein d'un client. Ceci peut donc créer des problèmes quant à la gestion d'un grand nombre de sessions.

AJAX se basant essentiellement sur l'échange asynchrone de données il est aisé de comprendre que l'échange des données entre le client et le serveur repose essentiellement sur l'identification d'une session.

Une session est une sorte de canal de communication 'unique' entre le client et le serveur. Lors de sa création, le serveur assigne un 'identifiant' au canal de communication entre le client et le transmet à chaque échange de données. Ce dernier est propre à chaque client et doit être unique pour que le serveur ne puisse pas confondre les différentes sessions qui sont en cours de traitement. Lorsque le client répond à une requête du serveur ce dernier transmet également cet identifiant dans chacune de ses réponses ou requêtes. Ainsi, l'échange entre plusieurs clients et un serveur peut être assuré.

3.2.1 Piratage de sessions

L'identifiant d'un canal de communication, étant le seul élément à pouvoir permettre au serveur de savoir avec quel client il communique, peut faire l'objet d'une faille de sécurité. En effet, si on imagine qu'une tierce personne puisse récupérer l'identifiant de cette session en interceptant par exemple les messages que se transmettent le client et le serveur[14], il suffirait ensuite à cette personne de jouer le rôle d'intermédiaire et simuler les interactions du client avec le serveur. Cette technique est communément appelée « Sessions Hijacking » ou « vol de session ».

[14] En utilisant un « sniffer ». Programme qui permet de capturer du trafic réseau. Sa description et son fonctionnement sont hors du sujet de ce travail. Il est également possible de fabriquer un sniffer avec Javascript et en injectant ce dernier via une faille XSS.

On peut imaginer qu'une banque, soucieuse de conserver son image de banque jeune et dynamique, souhaite offrir à ses clients des outils permettant de simplifier les tâches rébarbatives que sont la saisie du compte bancaire du destinataire pour les versements récurrents par exemple. L'application d'e-banking pourrait conserver en mémoire la liste des comptes bancaires sur lesquels le client effectue le plus souvent des versements. Lors de l'entrée du numéro du compte, l'application enverrait les premiers caractères du compte au serveur (au moyen d'un canal utilisant AJAX et identifié par un identifiant de session) qui chercherait dans sa mémoire le ou les comptes correspondants et renverrai une liste de suggestions au client qui pourrait facilement choisir le compte voulu sans avoir à terminer la saisie.

Cet exemple prouve qu'AJAX apporte une nouvelle fois son lot d'innovation en terme d'ergonomie, mais ouvre également une grosse brèche de sécurité dans l'application. Si l'on imagine que le client était victime d'un piratage de session, le pirate pourrait agir à l'insu du client et verser des sommes sur son propre compte ou autre compte frauduleusement obtenu.

3.2.2 XSS ou « Cross Site Scripting »

L'attaque XSS ou dite de « Cross Site Scripting » se situe au niveau du navigateur Internet. Elle consiste à déposer sur un site Internet des données arbitraires par exemple en laissant un commentaire sur un forum. Les données, sous la forme d'une expression HTML ou Javascript, si elles ne sont pas correctement filtrées peuvent êtres insérées sur le site victime. Par la suite, lors de son affichage, il est exécuté sur le navigateur de la victime et suivant la nature du code permet les actions suivantes :

> *« L'exploitation d'une faille de type XSS peut permettre à un intrus de réaliser les opérations suivantes :*
>
> *- Affichage d'un contenu non interne au site (publicité, faux article)*
>
> *- Redirection (parfois de manière transparente) de l'utilisateur.*
>
> *- Vol d'informations, par exemple sessions et cookies.*
>
> *- Actions sur le site faillible, à l'insu de la victime et sous son identité (envois de messages, suppression de données, etc.)*
>
> *- Plantage de la page (boucle infinie par exemple), et souvent du navigateur.*
>
> *Une faille de type XSS est à l'origine de la propagation des virus Samy ((en) explication et source du ver) sur MySpace en 2005 ainsi que de Yamanner sur Yahoo!Mail en 2006. »*

Source : http://fr.wikipedia.org/wiki/Cross_site_scripting

Le risque est donc réel pour l'utilisateur qui peut transmettre des informations sensibles à son insu.

Avec l'apparition d'AJAX ce genre d'attaque peut se révéler particulière perditieuse car la victime ne voit absolument rien (puisque AJAX s'exécute en tâche de fond) et aucune action n'est requise de sa part. Le chapitre « Deux géants se font avoir » détaille deux des attaques lancées sur des sites à fort trafic.

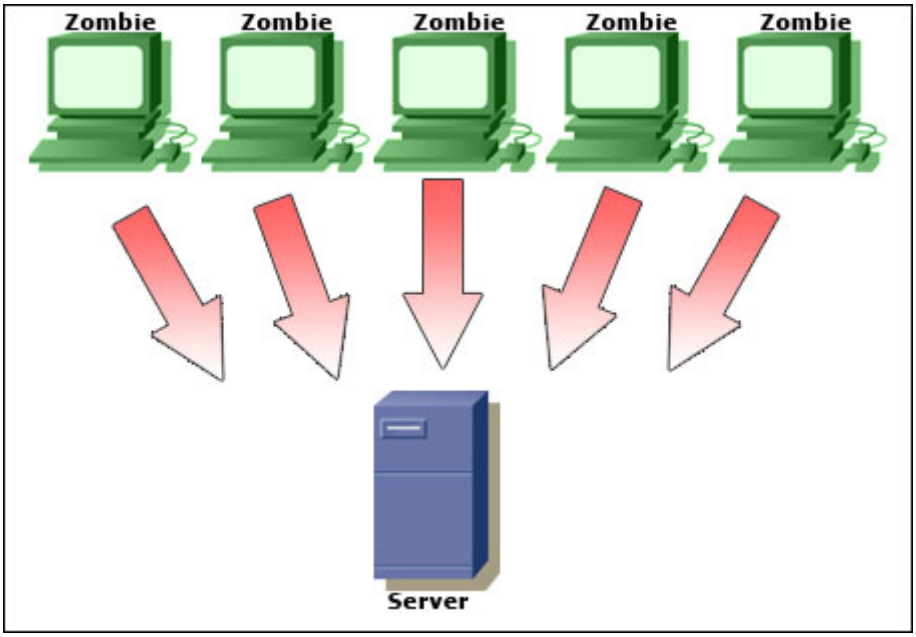
3.3 Déni de service

On parle de déni de service lorsqu'un ou plusieurs processus monopolisent l'ensemble des ressources d'un serveur. Pour un serveur Web on parle généralement de bande passante ou de ressources en terme de calcul du processeur.

On peut illustrer ce problème par la connexion simultanée de plusieurs machines en demandant l'accès à un service bien précis du serveur Web. Ces machines sont généralement des « Zombies » ou ordinateurs qui ont été piratés et sont contrôlés par des pirates.

Figure 8

Déni de service



Source : http://www.learn-networking.com/security/images/distributed-denial-of-service.jpg

Lors d'une attaque de type déni de service, il est très difficile voir impossible de reprendre le contrôle du serveur qu'il faut la plupart du temps redémarrer et changer son adresse IP.[15]

Quel rapport avec AJAX ?

L'utilisation d'AJAX augmente le trafic puisqu'il permet d'envoyer des données de manière asynchrone sans avoir besoin de l'intervention du client. Le fait d'autoriser l'accès aux pages qui sont utilisées pour recueillir les informations transmises par le client peut conduire le pirate à découvrir de nouvelles portes d'entrées au serveur cible et ainsi constituer une brèche de sécurité qui peut conduire à la monopolisation des ressources

Le danger peut aussi venir de l'intérieur. Un code Ajax mal contrôlé peut envoyer à son propre serveur des requêtes en boucle, créant ainsi un auto déni de service.

Avec AJAX le serveur se retrouve donc à gérer plusieurs sessions imbriquées : des sessions utilisateurs et les actions que l'utilisateur effectue au sein d'une session.

Cette augmentation du nombre d'éléments à gérer a pour première conséquence d'augmenter le trafic réseau. Si ce dernier n'arrive pas encore à saturation, on peut imaginer que dans quelques années l'utilisation massive de ce dernier pourra conduire à des congestions.

3.4 Deux géants se font avoir

A peine AJAX commençait-il à être utilisé que deux grands acteurs d'Internet subirent l'attaque de deux virus/vers qui exploitèrent une faille dans le filtrage des codes Javascript et permirent l'utilisation d'AJAX. C'était la première fois qu'un virus pouvait se propager sans avoir besoin de l'action d'un utilisateur.

3.4.1 Yamanner, virus, Yahoo mail

Le 12 juin 2006, les utilisateurs de Yahoo mail ont reçus un étrange message dans leur boite de messagerie. Un message avec pour sujet : « New Graphic Site » émis par av3@yahoo.com. Une fois le message ouvert, la propagation se faisait automatiquement aux autres contacts du carnet d'adresse du compte. Baptisé « Yamanner » par les différents éditeurs d'anti-virus il était le premier virus à cibler spécifiquement un service de messagerie Web, infectant aussi un poste de travail sans

[15] Vision très simplifiée, la réalité est un peu plus complexe.

qu'il ne soit nécessaire d'ouvrir une pièce jointe piégée embarquant du code malveillant, comme c'est couramment le cas. « Yamanner » aura contaminé en tout plus de 200 millions d'utilisateurs dans le monde.

3.4.2 Samy, ce héro !

Le site communautaire MySpace n'a pas été non plus à l'abri de l'exploitation de faille de sécurité dans le décodage et l'affichage d'images. Elle a également souffert de l'exploitation d'une faille du navigateur Internet.

En octobre 2006, un jeune homme de 19 ans a eu l'idée d'exploiter ces failles en ajoutant à sa page personnelle un script AJAX qui s'exécutait automatiquement lors de la visite de cette dernière. Ce script consistait à faire ajouter « Samy »[16] à la liste des amis du visiteur en faisant tourner en tâche de fond des confirmations d'ajout sans que le visiteur n'en soit informé ni qu'il ait besoin d'aucune action de sa part. L'exploitation de cette faille a permit à Samy de se faire plus d'un million d'amis en l'espace de 48 heures.

Cet exemple est un bon exemple d'attaque XSS et démontre à quel point il est vital de se prémunir contre cette dernière. Cette attaque n'avait pas de but malveillant mais on peut très bien imaginer un site marchand de commerce en ligne sur lequel un pirate aura écrit un commentaire en relation avec un article qu'il aurait testé en indiquant son expérience. Il pourrait inclure des instructions malveillantes en Java script qui s'exécuteraient en tâche de fond et permettraient au pirate de subtiliser des informations quant au visiteur et lui permettre par exemple d'accéder à son profil et de faire des achats à l'insu de la victime.

3.5 Vie privée

3.5.1 Espionnage à l'insu de l'utilisateur

L'échange de messages entre le serveur et l'utilisateur étant totalement transparent il est possible que ce dernier ne connaisse pas exactement la nature des données transmises au serveur. Ces dernières peuvent l'être à son insu. Il s'agirait de savoir jusqu'où cette technologie peut s'immiscer dans la vie privée des utilisateurs. L'exemple de l'utilisateur qui demande le remplacement de son baladeur mp3 sur un site marchand en ligne est un bon exemple.

[16] Nom du profil qui a servi de base à l'attaque

Imaginons que l'utilisateur soit en train de remplir le formulaire de demande de remplacement de son baladeur. Dans la première version de sa requête il noterai dans le champ texte correspondant sur le site du marchant que le baladeur est tombé dans les escaliers puis n'a plus fonctionné (clause que la garantie d'utilisation de l'appareil ne couvre évidement pas), puis, réalisant que peut-être cet argument ne jouerait pas en sa faveur déciderait d'effacer la mention de la chute dans les escaliers. Si on imagine que le site marchand pouvait savoir l'historique des caractères tapés dans le champ texte, il serait en mesure de connaître la vérité et ceci à l'insu de l'utilisateur ce qui peut constituer une intrusion dans la sphère privée.

A ce sujet le préposé fédéral à la protection des données édicte dans la Loi fédérale sur la protection des données du 15 décembre 2006 :

> *« Art. 4 Principes*
>
> *[1] Toute collecte de données personnelles ne peut être entreprise que d'une manière licite.*
>
> *[2] Leur traitement doit être effectué conformément aux principes de la bonne foi et de la proportionnalité.*
>
> *[3] Les données personnelles ne doivent être traitées que dans le but qui est indiqué lors de leur collecte, qui est prévu par une loi ou qui ressort des circonstances. »*
>
> Source : http://www.admin.ch/ch/f/rs/235_1/index.html

Le site a donc l'obligation légale d'avertir le client que des données sont d'une part collectées à son insu et qu'elles serviront peut-être à d'autres buts que ceux indiqués lors de leur collecte.

AJAX est une technologie très nouvelle et encore peu connue du public pour ses aspects techniques et ses possibilités. C'est la raison pour laquelle il n'existe pas vraiment d'avertissement sur les sites mais ce risque est bien réel.

De plus, les sites qui seraient hébergés en dehors du territoire Suisse pourrait rendre le problème de résolutions de différents et des poursuites très complexe du point de vue légal au même titre que c'est le cas pour les problèmes de piratage MP3 ou de pédophilie sur Internet.

4. Solutions

Les problèmes énoncés dans ce travail sont heureusement résolubles pour la plupart. Des voies

4.1 Protection de la vie privée

A titre d'avertissement il serait souhaitable que les sites qui utilisent AJAX à d'autres fins que la collecte stricte des données dans le but d'atteindre un but fixé comme l'achat d'un bien sur Internet mentionnent clairement au moyen d'un texte informatif que les données recueillies peuvent servir à des fins d'étude de comportement.

Un autre moyen à envisager serait de permettre à l'utilisateur de connaître la nature des données qui sont transmises à son insu. On peut penser créer une extension au navigateur qui informerait l'utilisateur des données qui transitent. Cette idée est traitée en détail dans le *chapitre 5*.

4.2 Gestion des sessions

Afin de contrer le problème du piratage de session il existe principalement deux solutions :

- Le cryptage des données entre client et serveur

  Ce dernier utilise une clef d'encryption entre client et serveur qui permet de communiquer de manière sécurisée. Cet encodage est appelé SSL (**S**ecure **S**ocket **L**ayer) et est utilisé lorsque le client doit transmettre des informations dites 'sensibles' au serveur.

  Ce protocole est fiable à 99% mais n'est pas totalement infaillible, elle est aussi susceptible d'être exploitée par une attaque du type 'homme du milieu' comme l'attaque du « Session Hijacking » une tierce personne peut se faire passer pour le serveur et faire croire au client qu'il est bien en communication avec le serveur réel. Décrire ce type d'attaque sort du cadre de ce travail, mais le lecteur pourra se référer à l'*annexe 1* du présent travail pour plus d'informations.

- L'authentification dite « forte » du client

  Une sécurité supplémentaire que l'on pourrait imaginer serait d'utiliser une méthode d'authentification qui serait physiquement séparée de la

machine du client. L'utilisateur disposerait d'une calculette sur laquelle il pourrait entrer un code transmis par le serveur et cette dernière retournerait une réponse qui serait propre à cet utilisateur. De ce fait, il serait impossible pour un attaquant d'imiter la réponse du client puisque cette dernière serait inconnue jusqu'à la réponse de la calculette.

4.3 Une bonne Implémentation : quelques règles

La véritable clef du succès d'une bonne implémentation et utilisation d'AJAX est le respect stricte de plusieurs règles de base à sa voir :

4.3.1 Schémas d'authentification

Ne pas laisser l'accès libre à n'importe quel script qui serait accessible en dehors d'un schéma d'authentification stricte. Tout utilisation d'un script AJAX qui se connecterait à un script devrait se faire au travers d'un schéma d'authentification, aucune action ne devrait être permise sans la création préalable d'une session définie pour un utilisateur.

Afin de protéger un accès, le concepteur pourrait créer un module de connexion à n'importe lequel de ses scripts qui sont utilisés par le client via des requêtes asynchrones. Le module se chargerait de vérifier la validité des requêtes et serait en mesure de filtrer les requêtes d'origine douteuses, par exemple des milliers de connexions au même script provenant de la même adresse IP déclencherait une alerte.

On pourrait imaginer un système de détection d'intrusion à la manière des pare-feux modernes qui sont capable de reconnaître des signatures d'attaque et de les bloquer avant même qu'elles atteignent les serveurs. Cette double sécurité protègerait l'application des failles classiques. Le plus sûr serait bien évidemment de ne pas utiliser AJAX lorsque cela est possible.

4.3.2 Ne jamais faire confiance au client

Comme bonne règle de base on peut appliquer la politique de la « tolérance zéro » qui consiste à n'absolument pas faire confiance aux entrées de l'utilisateur. Par exemple lors du remplissage d'un formulaire en ligne il ne faudrait jamais partir du principe que l'utilisateur entrera des données valides et toujours tester et valider ces dernières avant de les entrer dans la base de données du système d'information.

Ce principe peut être très contraignant du point de vue du programmeur qui doit vérifier que tous les points d'entrée de son application soient effectivement validés mais

permettent de ne pas laisser la possibilité aux utilisateurs malveillants de pouvoir introduire des données qui pourraient endommager le système. A ce titre, on peut imaginer une application qui ne filtre absolument pas la taille des données fournies par un client lors de son enregistrement. Si ce dernier était malveillant, il pourrait entrer autant de données qu'il le souhaite jusqu'à pouvoir saturer complètement la mémoire du serveur de l'application et le rendre ainsi inutilisable. On peut aisément comprendre l'enjeu en terme de disponibilité et donc d'argent que cela remettrait en cause pour l'entreprise qui fournirait ce service.

4.3.3 Logique business sur le serveur

Il est absolument primordial de ne jamais implémenter une quelconque forme de logique business chez le client. Par exemple le calcul du total d'un panier d'achat sur un site marchand ne devrait se faire que du côté serveur. Il serait bien malheureux de laisser ce calcul uniquement au client qui pourrait changer la valeur du total à sa guise. Les versements étant pour la plupart automatisés il serait fâcheux de constater que le vendeur aura vendu tout son stock pour rien.

4.3.4 Légitimité des requêtes

Comme décrit au point 4.3.2, on peut également observer d'où proviennent les requêtes et mettre en doute leur légitimité. Des requêtes provenant d'une adresse IP qui ne se serait pas connectée au site au travers d'une authentification serait douteuse.

4.3.5 Validation des données

La validation des données est souvent laissée pour compte par le concepteur. Cette dernière, faute de temps, reste la plupart du temps côté client. Cette pratique est à bannir de tous les bons manuels de programmeurs car elle mettrait en péril toute la sécurité de l'application. Par exemple la validation d'un nom d'utilisateur qui ne devrait pas comporter de caractères spéciaux devrait se faire uniquement sur le serveur[17]. Le client n'étant par définition par 'fiable' il est important de respecter cette consigne. Le non respect de cette règle peut, dans certaines circonstances permettre l'insertion non autorisée de données directement dans le système d'information du site. Un exemple détail cette attaque au *chapitre 5*.

[17] On peut toutefois implanter ce test côté client à condition bien sûr de l'implanter quoi qu'il arrive sur le serveur qui reçoit la requête.

4.4 Du côté des navigateurs

Le navigateur est bien sûr la clé de voûte de la sécurité côté client. Outre le fait qu'il faille faire attention à la programmation les navigateurs ne sont pas eux non plus à l'abri de l'exploitation de failles. Beaucoup de failles côté navigateur ont permis à des pirates de les exploiter. Il serait trop long de les décrire ici mais ce qu'il faut retenir comme règle est que le client devrait toujours avoir la dernière version de son navigateur.

4.5 Aide à la conception

Afin de ne pas toujours réinventer la roue le concepteur peut se baser sur des solutions déjà existantes. Ces dernières sont avantageuses pour les raisons suivantes :

- Solution éprouvées et déjà adoptée par des millions de sites.
- Grande communauté d'aide en cas de problème.
- Evolutivité garantie puisque maintenant par une grande communauté.

Il s'agit des frameworks Javascript qui permettent d'aider le concepteur.

4.5.1 Utilisation d'un framework existant

Au vu du grand succès d'AJAX beaucoup de frameworks de standardisation de son utilisation de ce dernier a vus le jour. Ces derniers intègrent les fonctionnalités et les modèles les plus couramment utilisés et évitent donc au client de devoir « réinventer la roue ».

Voici les caractéristiques de quelques-uns d'entre eux :

4.5.1.1 jQuery

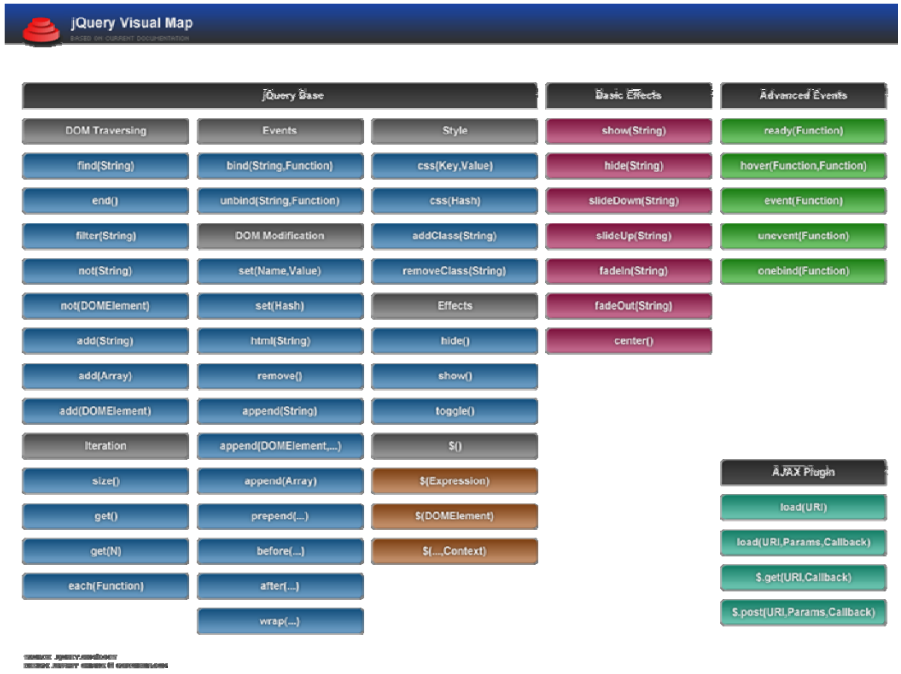
- Léger - 14Ko compressé
- Compatible avec Internet Explorer 6.0 et supérieur, Firefox 1.5 et supérieur, Safari 2.0 et supérieur, Opera 9.0 et supérieur.
- Puissant - avec la possibilité d'effectuer plusieurs actions sur un élément grâce au chaînage de fonctions

- Non intrusif - avec la possibilité de définir des évènements en dehors du contenu de la page grâce à à l'utilisation des sélecteurs CSS 3.

L'architecture des classes de jQuery est illustré sur la figure 9.

Figure 9

jQuery Framework



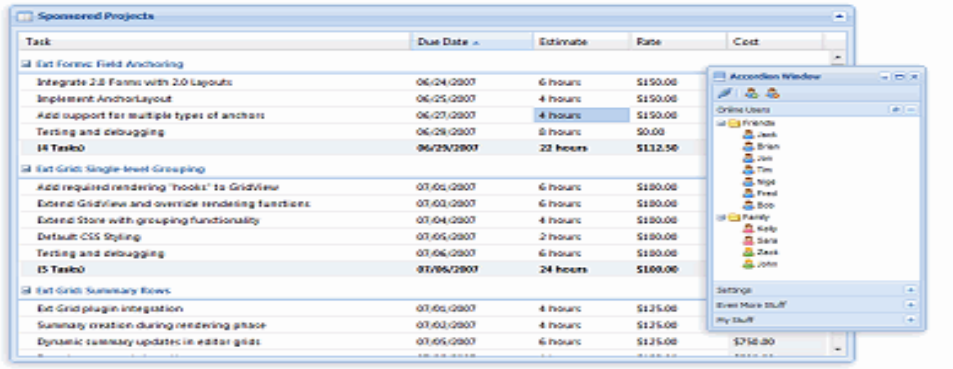
Source : http://ajax.phpmagazine.net/2006/04/jquery_visual_map.html

4.5.1.2 ExtJS

- Une interface graphique très évoluée
- Permet de créer des fenêtres modales, non modales
- Intègre des 'modules' comme l'interfaçage JSON ou XML
- Gère les tableaux, fenêtres, formulaires, …

Un exemple de fenêtre présentée à l'aide d'ExtJS est présenté sur la figure 10.

Figure 10
ExtJS Framework



4.5.1.3 Autres

Il est à noter qu'il existe beaucoup d'autres frameworks qu'il serait bien trop long de détailler ici. Il existe donc entre autre :

- Dojo
- Prototype
- Mootools
- Qooxdoo
- Script.aculo.us

4.5.2 Frameworks – gage de sécurité

Outre le fait de faciliter la création d'applications avec l'automatisation et la procédurisation des méthodes et actions les plus souvent effectuées par le concepteur.

L'utilisation d'un framework garanti que les problèmes de sécurité ont d'une part très peut de chance d'exister car un framework est en général le fruit de la collaboration de plusieurs centaines de programmeurs ce qui réduit donc considérablement le risque d'une faille due à une omission. D'autre part, ces frameworks possèdent chacun leurs propre communauté d'utilisateurs qui se comptent en général en centaine de milliers. De ce fait, le temps qui s'écoule entre la découverte d'une faille de sécurité et sa correction peut être très rapide.

Par opposition au concepteur qui est seul à construire sa propre architecture, l'utilisation d'un framework rends la tâche plus aisée et surtout plus sûre. Petit bémol toutefois à l'implantation du framework et des schémas d'authentification qui eux ne sont pas dépendants de l'utilisation d'un framework.

5. Démonstration d'une attaque

On peut édicter une méthodologie à appliquer pour trouver les points faibles d'une application dans le but de les exploiter. En imaginant être le pirate on chercher tout d'abord dresser une liste des points d'entrées qui sont possibles à l'attaque de l'application.

Les pages d'erreurs des navigateurs peuvent être un bon point de départ car elles indiquent souvent quels sont les programmes qui sont exécutés par l'application ce qui peut orienter l'attaque vers la recherche de failles de tel ou tel système d'exploitation.

Une fois les points d'entrées trouvés, le pirate s'attarde à regarder de plus près les interactions qui sont effectués par les appels AJAX entre client et serveur et peut juger de l'intérêt d'un appel à une page plutôt qu'un autre. Typiquement entre deux requêtes, une pour avoir la date de la dernière connexion et une autre concernant l'authentification de l'utilisateur, il n'est nul doute que le pirate exploitera celle de l'authentification en premier.

Dans un troisième temps, le pirate essaiera d'envoyer des données invalides pour observer la réponse de l'application, si déjà à ce niveau l'application permets l'exploitation de failles il ne sera pas nécessaire de continuer l'analyse. Le pirate exploitera donc une faille en injectant du code Javascript ou SQL si la faille le permet.

Entre temps, suivant la réponse du serveur et le temps que ce dernier mets à répondre le pirate pourrait trouver des failles qui concernent le traitement et dirigera ses attaques de déni de services aux fonctions qui sont les plus lourdes en terme de temps de calcul pour le serveur.

Enfin, si les points d'entrées de l'application ne sont pas exploitables, le pirate se tournera vers le code qui est exécuté par le client et cherchera les failles de programmation. Ces dernières étant côté client il est très aisé pour le pirate de les modifier, par exemple toute logique qui serait exécutée côté client serait facile à changer de manière à faire exécuter n'importe quel bout de code.

On voit donc bien que le fait de connaître les techniques d'un pirate peut aider à la sécurisation de l'application Web.

6. Aide à la sécurisation d'AJAX

Tous les problèmes de sécurité et de vie privée décrits plus haut nous amènent à nous poser la question de savoir s'il serait possible d'aider à sa sécurisation et si possible en automatisant ce procédé. On peut imaginer une aide pour le concepteur ou le prestataire de services Internet et une pour l'utilisateur final de l'application Web

6.1 Outil d'analyse syntaxique

Le principe d'un analyseur de failles de sécurité AJAX basé sur la recherche syntaxique reposerait sur le fait qu'il est facile d'identifier les requêtes Javascript qui permettent l'ouverture d'un canal de communication asynchrone entre client et serveur. On sait par exemple que toute requête *XMLHttpRequest* permet l'ouverture d'un canal asynchrone. Même en obfusquant[18] le script, cette instruction doit rester en clair pour être interprété par le navigateur. Il serait donc possible de faire un parseur de code à la recherche de ces dernières. Au vu de la complexité des applications AJAX cet outil se bornerait à afficher au concepteur le ou les classes qui font un appel l'objet de création d'un canal asynchrone avec AJAX pour lui permettre d'en vérifier la sécurité, il ne serait par contre pas possible de rendre automatique cette vérification puisque bien trop complexe à mettre en œuvre, le cerveau humain restant la seule méthode valide à ce jour.

6.2 Proxy applicatif

Une autre alternative pour la vérification de la sécurité des scripts utilisant AJAX serait de mettre en place un Proxy[19] qui pourrait être mis à la dispositions des concepteurs qui souhaiteraient tester la sécurité de leurs applications. Ce dernier pourrait analyser les requêtes et définir lesquels proviennent de requêtes AJAX et pourrait être programmé pour ajouter et introduire toute une panoplie de textes qui seraient là pour vérifier la bonne validation des entrées ou de tester des attaques connues du type d'injection SQL dans les données qui transiteraient.

Ce service pourrait être proposé par un prestataire de service Internet qui pourrait mettre en avant que la sécurité des applications qu'il héberge le préoccupe.

[18] Cachant de manière à rendre très peu lisible mais toujours exécutable

[19] Un Proxy est un programme qui agit comme une passerelle entre un client et un site Internet.

6.3 Extension navigateur

Du côté de l'utilisateur on pourrait imaginer un petit plugin ou extension au navigateur Internet qui serait en mesure de prévenir l'utilisateur qu'il est en train de transmettre des données asynchrones et ceci malgré que le chargement de la page soit terminé.

Un petit indicateur de couleur indiquerait à l'utilisateur si oui ou non il est en train de transmettre des données au serveur et au moyen d'un simple clique il serait en mesure de voir le détail des informations qu'il transmets et de les stopper ou voir de les interdire à la manière d'un bloqueur de popups actuellement.

7. Conclusion

Avec l'arrivée du Web 2.0 les internautes ont radicalement changé leur manière de communiquer. Internet est devenu un espace communautaire où des millions de personnes sont connectées non seulement sur la toile mais tissent des réseaux sociaux. L'utilisation d'AJAX et l'évolution des applications Web va sans nul doute se poursuivre et permettre à tout un chacun de profiter de nouveaux services de plus en plus interactifs.

Au vu de son évolution on pourrait tout à fait envisager qu'un jour il ne soit plus nécessaire de disposer d'un ordinateur puissant pour profiter pleinement de toutes les technologies qui seront proposées. Il suffirait alors d'avoir une connexion haut débit et un navigateur Internet. La société Google est en train de travailler à la réalisation d'un système d'exploitation entièrement basé sur Internet dans le plus grand des secrets.

AJAX aura bouleversé notre manière de concevoir une application Web. Il n'est nul doute que les grands acteurs d'Internet vont devoir travailler à sa sécurisation. Il sera peut-être judicieux de créer un nouveau langage qui soit aussi simple à utiliser que Javascript et qui intègre les des méthodes de communications asynchrones sécurisées de facto.

On constate qu'aujourd'hui la sécurité des échanges ainsi que la sûreté des données qui sont hébergées pose des questions auxquels il est difficile de répondre. L'augmentation du nombre d'applications utilisant AJAX est phénoménale mais le contrôle des failles est, peut-être relégué, au second plan car il faut désormais réagir vite au changement, notamment ceux de la concurrence.

Il est difficile pour l'heure de garantir une sécurité absolue, et l'utilisation d'AJAX peut être à double tranchant puisqu'il y a plus de choses à vérifier et à garantir contre une éventuelle intrusion.

Du côté de l'utilisateur, il semblerait que ce dernier se borne à profiter d'un gain d'interactivité qu'AJAX apporte à ses applications sans trop se soucier des éventuels problèmes qu'il pourrait occasionner en autre pour le respect de la vie privée. Il y a certes quelques outils pour aider le client à mieux comprendre ce qui se passe à l'envers du décor comme une extension pour Firefox (et donc pas pour Internet Explorer qui détient 80% du marché) dénommée « FireBug » permets d'analyser les requêtes qui sont envoyées au serveur avec AJAX mais son utilisation reste très complexe pour l'utilisateur néophyte.

La vie privée quant à elle reste et restera toujours un éternel problème sur Internet. Jusqu'où les sites pourront aller pour dévoiler la vie privée de leurs membres ?

Une anecdote parue il a quelque jours illustre parfaitement cette problématique :

> *« Mercredi 31 octobre 2007, le jeune Kevin Colvin, stagiaire à l'Anglo Irish Bank, a prévenu ses supérieurs qu'il serait absent le lendemain pour «raisons familiales». Le 1er novembre, son patron lui a envoyé sa réponse : «Merci de nous avoir prévenu. J'espère que tout va bien à New York. (Sympa la robe).» En pièce jointe, une photo de Kevin déguisé en fée en train de fêter Halloween. Son patron l'a trouvée quelques heures après la soirée sur le site communautaire Facebook. »*
>
> Source : http://lefigaro.fr, 07/12/2007

L'évolution très rapide des applications Web permet difficilement d'avoir assez de recul pour juger des risques que l'utilisation d'AJAX aura créé. Il sera sans doute bon d'attendre encore quelques années afin d'en tirer un vrai bilan.

8. Bibliographie

http://www.zdnet.fr/actualites/imprimer/0,50000200,39286339,00.htm
http://www.cgisecurity.com/ajax/
http://www.readwriteWeb.com/archives/fear_of_Web_20.php
http://www.enterprisenetworksandservers.com/opinionw/art.php?291&t=1191237633
http://www.acunetix.com/Websitesecurity/ajax.htm
http://www.securityfocus.com/print/infocus/1868
http://www.spidynamics.com/assets/documents/AJAXdangers.pdf
http://openajax.org/member/wiki/AJAXWorld_2007_September_Ajax_Security_Iues
http://journaldunet.com/solutions/0606/060615-securite-yamanner-virus-ajax.shtml
http://www.devx.com/Webdev/Article/28861/1954?pf=true
http://whitehatsec.com/home/resources/articles/files/myth_busting_ajax_insecurity.html
http://www.net-security.org/article.php?id=949
http://www.securityfocus.com/infocus/1879
http://blogs.zdnet.com/Web2explorer/?p=285
http://fr.wikipedia.org/wiki/Rich_Internet_Application
http://ajaxian.com/archives/sprajax-an-ajax-security-scanner
http://www.lemondeinformatique.fr/actualites/lire-ajax-une-nouvelle-generation-de-virus-multiplateformes-19789.html
http://www.edoeb.admin.ch/dokumentation/00445/00509/00512/00803/index.html?lang=fr
http://searchsoa.techtarget.com/qna/0,289202,sid26_gci1164745,00.html
http://www.securityfocus.com/infocus/1868/2#ref3

Annexe 1
Les attaques contre le protocole SSL

Dans cette section, nous allons tenter de vous décrire les différentes attaques qui pourraient être utilisées contre le protocole SSL. Nous ne pouvons cependant pas garantir l'exhaustivité de cette liste. SSL a été défini pour contrer ces attaques.

Craquer les Ciphers

SSL dépend de plusieurs technologies cryptographiques différentes. Le chiffrement par clé publique de RSA est utilisé pour l'échange de la clé de session et pour l'authentification du client/serveur. Différents algorithmes cryptographiques sont utilisés. Si des attaques cryptographiques sont réalisées avec succès contre ces technologies, le protocole SSL ne pourra plus être considéré comme sûr.

Des attaques contre une session de communication spécifique peuvent être réalisées en enregistrant la session et ensuite utiliser cet enregistrement pour craquer soit la clé de session, soit la clé publique RSA jusqu'à l'obtention de la communication claire. Cet approche est plus facile que de craquer les technologies cryptographiques pour tous les messages possibles. Notez que SSL tente de rendre le coût d'une telle attaque supérieur au bénéfice d'une attaque réussie, l'attaque devient donc une perte d'argent et/ou de temps.

Attaque à "texte clair"

Les attaques à "texte clair" ont lieu lorsque l'attaquant a une idée du message chiffré qui est envoyé. L'attaquant peut générer une base de données dont les clés sont la valeur cryptée du texte clair et dont les valeurs sont la session cipher key. Lorsque cette base de données est construite, une simple fonction de recherche identifie la clé de session qui va avec une valeur chiffrée particulière. Une fois la clé de session connue, le message entier peut être déchiffré. Un hardware adéquat peut rendre ce système peu coûteux et très rapide.

A cause de la nature même de SSL, les attaques à "texte clair" sont possibles. SSL tente tout de même d'éviter ce genre d'attaque en utilisant des clés de session très grandes. D'abord, le client génère une clé plus grande que permise par exportation, et envoie une portion de celle-ci en clair au serveur. Cette portion claire de la clé, concaténée avec la portion secrète donne une clé très grande.

Le hardware nécessaire pour réaliser une telle attaque est donc rendue prohibitive. Chaque bit ajouté à la longueur de la clé double la grandeur de la base de données. En utilisant une clé d'une longueur de 128 bits, la base de données nécessaire n'est pas réalisable. Même si une base de données plus petite est utilisée, elle doit d'abord être générée avec les bits clairs. Il s'agit alors d'un processus très lent et coûteux.

Une conséquence de la défense SSL est que les attaques de force brute deviennent les attaques les moins chères. Les attaques de force brute ont un rapport espace/temps bien connu, et il devient alors possible de déterminer le

coût d'une attaque. Pour une clé à 128 bits, le coût est théoriquement infini. Pour une clé à 40 bits, ce coût est fortement réduit mais toujours hors de portée d'un "random hacker".

Attaque replay

Une attaque replay est simple. Une personne enregistre une session de communication entre un client et un serveur. Plus tard, cette personne se connecte à ce même serveur et rejoue les messages du client qu'il vient d'enregistrer.

SSL élimine cette attaque en utilisant une "nonce" (connection-id) qui est "unique" pour chaque connexion. En théorie, le malfrat ne peut prédire à l'avance la nonce car elle est basée sur une série d'événements sur lesquels il n'a aucun contrôle. Le malfrat ne peut donc pas répondre aux demandes du serveur.

Une personne avec un bon matériel peut enregistrer beaucoup de session entre un client et un serveur et tenter de choisir la bonne session en se basant sur la nonce que le serveur envoie initialement dans son message SERVER-HELLO. Cependant, les nonces SSL ont une longueur de 128 bits. Le malfrat aurait donc besoin d'environ 264 nonces pour avoir 50% de chance de choisir la bonne session. Ce nombre est suffisamment grand que pour rendre le coût du matériel nécessaire à l'enregistrement prohibitif.

L'homme au milieu

L'attaque de "l'homme au milieu" peut avoir lieu lorsque trois personnes sont dans une session de communication : le client, le serveur et le malfrat. Ce dernier se situe sur le réseau entre le client et le serveur et intercepte le trafic provenant à la fois du client et du serveur.

L'homme du milieu opère en faisant semblant d'être le vrai serveur envers le client. Avec SSL, cette attaque est impossible grâce à l'utilisation de certificats du serveur. Durant la poignée de main initiale, le serveur est demandé à présenter un certificat signé par une autorité. La clé publique du serveur, son nom ainsi que le nom du donneur du certificat sont contenus dans le certificat du serveur. Le client vérifie le certificat en regardant la signature et en vérifiant si le nom du donneur de certificat est quelqu'un que le client reconnaît.

De plus, le serveur doit chiffrer quelque chose avec la clé privée qui s'associe avec la clé publique mentionnée dans le certificat. Seul un serveur qui possède à la fois le certificat et la clé privée peut répondre convenablement à ce challenge.

Si l'homme du milieu fournit un faux certificat, la vérification de la signature le détectera. Si la signature est une signature légitime pour le malfrat mais pas pour le serveur, la signature passera mais alors c'est la vérification du nom du serveur qui détectera l'effraction.

Finalement, si le malfrat fournit un certificat légitime pour le serveur en question, la signature et le nom passeront. Cependant, le malfrat, ne possédant pas la clé privée du serveur, ne pourra pas encoder convenablement sa réponse au challenge et sera détecté lors de cette dernière vérification.

Dans le cas très peu probable où le malfrat a deviné la clé privée, il ne peut toujours pas déchiffrée la clé de session et ne peut donc pas visualiser l'information chiffrée.

Source : http://membres.lycos.fr/sakajimo/ssl_img.htm