UE MIF30

Compte Rendu

Gestion des identités en ligne avec OpenID

KENFACK Patrick 10204072

Dirigé Par : **Yves Gérard**

SEMESTRE D'AUTOMNE 2008/2009

INTRODUCTION

La gestion des identités numériques est un sujet très actif aujourd'hui, qui a donné lieu à de nombreuses études et à plusieurs mises en œuvre. Au sein de ce vaste domaine, **OpenID** est une proposition d'attribution d'un URL comme identité, et d'un mécanisme pour authentifier Cette identité. Le fournisseur d'identité est séparé du vérificateur. Avec **OpenID**, un site Web Peut authentifier un client sans que celui-ci ait besoin d'un compte sur le site Web, et sans Qu'il existe de mécanisme central de gestion des identités.

Avant de définir et d'analyser un **OpenID**, il est important de définir une notion importante à savoir : **l'identité** qui permet d'authentifier une personne ou quelque chose.
L'identité étant un thème récent, il en existe d'autres approches. Une des plus connues est celle popularisée par Kim Cameron, architecte Identité chez Microsoft et connue sous le nom des « Sept lois de l'identité».

Cameron présente l'identité comme une suite d'affirmations comme « Cet étudiant a la carte d'étudiant n° 234-12-81 » ou bien « Kim Cameron est de nationalité états-unienne » ou encore « Cette personne a plus de dix-huit ans ». Ces affirmations peuvent éventuellement être étayées, c'est l'authentification. Cameron critique fortement les systèmes centralisés, y compris le projet Passeport de sa société, Il plaide au contraire pour un méta système d'identité, qui pourrait fédérer des systèmes divers, gérés par des organismes différents et proposant des services différents. Ses principes sont résumés en sept lois :

1. Contrôle par l'utilisateur. La nécessité du consentement éclairé est posée en principe. Notons que Cameron, malgré l'utilisation qu'il fait du terme de « loi », n'envisage pas de traduction législative de ses principes. Cette première loi ressemble beaucoup à des lois comme Informatique et Libertés en France.

2. Divulgation minimale. Reprenons l'exemple de l'affirmation comme quoi le sujet a plus de dix-huit ans. Pour Cameron, devoir montrer un document d'identité pour prouver son âge dans un bar est excessif car ce document donne plus d'informations que celle qui compte, le fait d'être majeur.

3. Divulgation uniquement aux entités qui en ont besoin. Ainsi, que Google contrôle l'identité de ceux qui accèdent à Gmail est raisonnable mais on ne voit pas pourquoi l'identité Google devrait être utilisée par une autre société.

4. Identité dirigée. Certains systèmes d'identité diffusent à tous (c'est le cas des passeports RFID et des passes Navigo, facilement lisibles à distance). Parfois, c'est souhaitable (il existe des services à vocation publique), parfois non et un système d'identité devrait être directionnel.

5. Pluralisme des acteurs et des techniques. Tirant les leçons de l'échec de Passport, Cameron affirme qu'il n'y aura pas de système unique.

6. Prise en compte des facteurs humains. Cameron fait remarquer que des techniques excellentes sur le papier, comme l'utilisation de certificats X509 pour authentifier

Les sessions SSL, ont échoué pour des raisons non techniques, par manque de prise en compte des utilisateurs (« Voulez-vous vraiment autoriser ce certificat signé par une autorité inconnue? »).

7. Maintien d'une certaine cohérence, malgré la variété des acteurs et des technologies. Par exemple, ses différentes identités doivent, pour l'utilisateur, apparaître de manière similaire (ce que font les cartes de Cardspace).

Cependant, ayant donné une définition de ce que c'est l'identité nous allons maintenant définir **qu'est ce qu'un OpenID.**

1. Définition

1.1. Selon wikipédia

OpenID est un système d'authentification décentralisé qui permet l'authentification unique, ainsi que le partage d'attributs. Il permet à un utilisateur de s'authentifier auprès de plusieurs sites (devant supporter la technologie) sans avoir à retenir un identifiant pour chacun d'eux mais en utilisant à chaque fois un unique identifiant **OpenID**.
Le modèle **OpenID** se base sur des liens de confiance préalablement établis entre les fournisseurs de services (sites web utilisant **OpenID** par exemple) et les fournisseurs d'identité (**OpenID** providers). Il permet aussi d'éviter de renseigner à chaque fois un nouveau formulaire en réutilisant les informations déjà disponibles.

1.2. Autre définitions

OpenID est une solution libre et décentralisée d'authentification unique. Elle vous permet d'obtenir rapidement une identité numérique, de changer ou de révoquer cette identité tout aussi rapidement. L'architecture étant décentralisée, vous ne dépendez pas que d'un seul fournisseur de service : vous pouvez en changer régulièrement et facilement ou bien carrément héberger vous-même votre identité numérique ! Grâce à votre identité numérique **OpenID**, vous pourrez :
Vous connecter une seule fois et accéder à tous vos sites préférés sans retenir tous les couples identifiant/mot de passe auparavant nécessaires ;

Centraliser les modifications de vos informations (exemple : changement d'adresse email...) ;
Gérer les autorisations d'accès à vos informations de chaque site visité.
Attention, il n'est pas question ici de sécurité. D'un point de vue du site qui demande une authentification, **OpenID** permet uniquement de confirmer que la personne qui souhaite se connecter est bien la personne dont l'identité numérique est décrite par l'URL fournie, et que les informations fournies par le serveur **OpenID** concernent bien cette personne. À partir de là, il y a deux cas de figure :

- soit le site a une confiance aveugle au serveur **OpenID** et décrète que si l'utilisateur est valide, alors il peut se connecter au site ;
- soit le site demande l'authentification au serveur **OpenID**, mais gère ensuite lui-même les droits d'accès. **OpenID** n'est qu'une brique simplifiant l'authentification. Elle ne dispense pas de mettre en place une politique de sécurité sur les sites utilisant cette technologie.

2. Fonctionnement d'un OpenID

Cependant, pour bien comprendre le fonctionnement d'un **OpenID**, il est nécessaire de bien distinguer les différents acteurs qui participent au bon déroulement du processus de mise en œuvre du mécanisme de fonctionnement des **OpenID**.

2.1. Acteurs

On distingue :

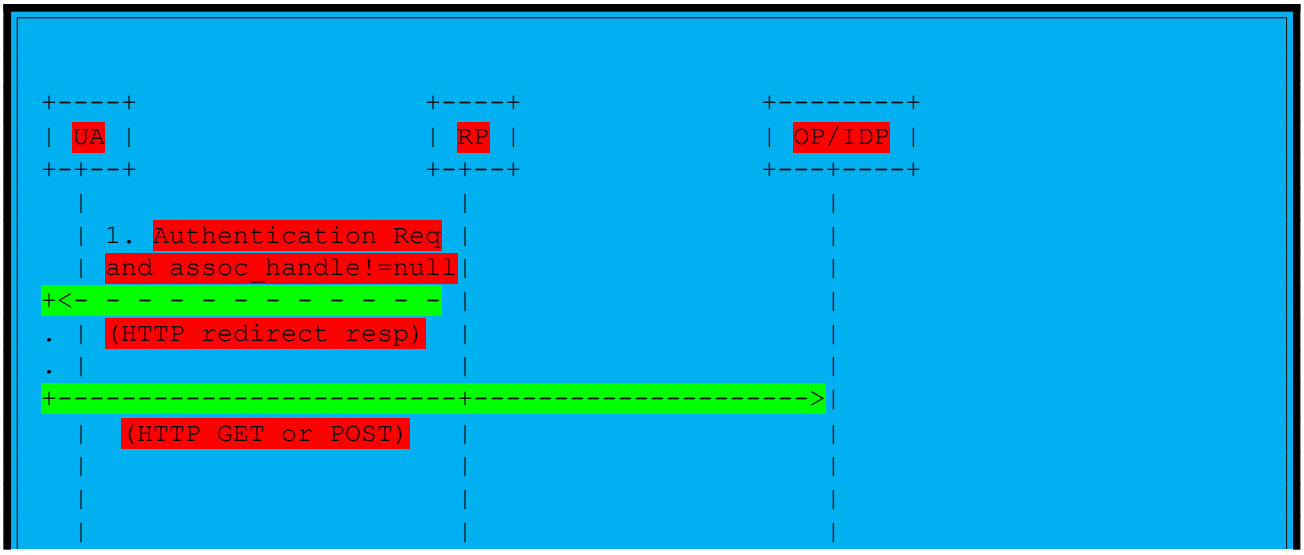
- l'utilisateur
- le consommateur (en abrégé, **RP**, **pour Relaying Party**) : il désire avoir une preuve de L'identité de l'utilisateur. C'est un site web, un service web, …
- le fournisseur d'identité (**OpenID Provider, ou OP, ou IdP, ou IP ou juste Provider**) : c'est un serveur d'authentification **OpenID** qui est contacté par le consommateur pour obtenir Une preuve de l'identité de l'utilisateur ;
- le client, par exemple le navigateur internet de l'utilisateur.

2.2. Principe

Lors d'une authentification unique, et donc lors d'une authentification via **OpenID**, on trouve trois entités : l'utilisateur qui souhaite s'authentifier (en fait, le navigateur utilisé, EU ou End User) ; le site sur lequel l'utilisateur souhaite s'authentifier (RP ou Relaying Party) ; le serveur d'authentification (OP ou **OpenID** Provider, ici le serveur **OpenID**).

Lors de cette authentification, les échanges suivants ont lieu entre les trois entités : l'utilisateur entre son identifiant **OpenID** sur le site sur lequel il souhaite s'authentifier ; le site contacte alors le serveur, et ils partagent un secret ; l'utilisateur est redirigé vers le serveur qui lui demande de s'authentifier (s'il s'agit du premier accès de la session de navigation) ; l'utilisateur est averti des données auxquelles le site souhaite accéder ; l'utilisateur décide des données qu'il souhaite communiquer au site ; l'utilisateur est alors redirigé vers le site, avec ses informations cryptées grâce au secret partagé établi entre le site et le serveur ; l'utilisateur est authentifié sur le site.

L'utilisateur a donc la main sur ce qu'il souhaite communiquer au site, et ces informations sont transmises de manière sécurisée du serveur vers le site.
Le tableau ci-dessous montre comment se mécanisme d'authentification établie la mise en association avec les différents acteurs intervenants dans ce mécanisme.



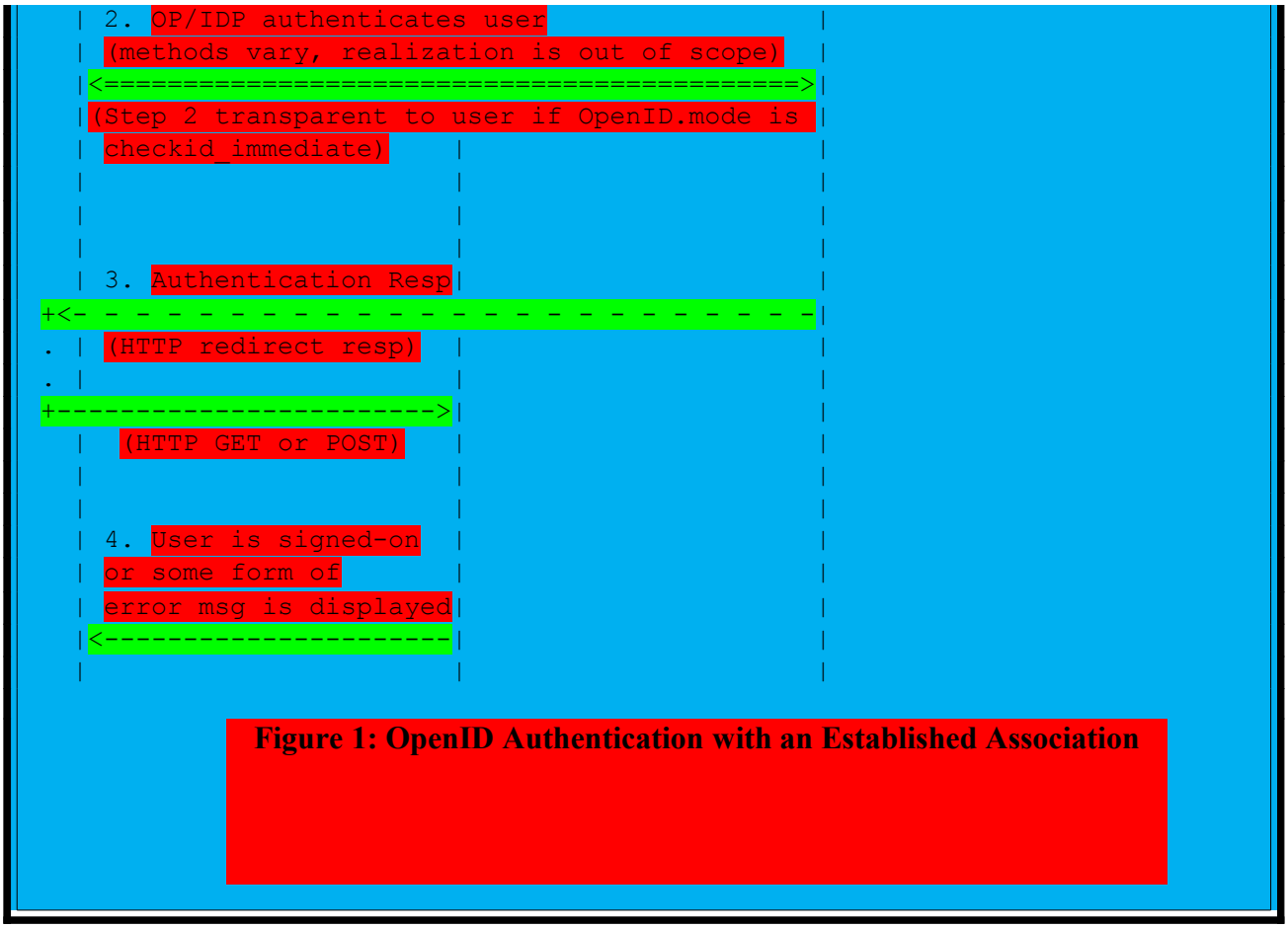


Figure 1: OpenID Authentication with an Established Association

Cependant on constate que les spécifications d'**OpenID** couvrent le protocole de manière globale, car elles sont spécialisées par le biais d'extensions :

- Simple Registration Extension (SREG) : elle permet un échange de profil très léger, adapté à l'utilisation sur la plupart des sites, ce profil léger ne contenant que les 9 champs identifiés comme récurrents pour les identifications web (pseudonyme, nom complet, adresse email, date de naissance, langue, horaire, sexe, pays et code postal).
- Yadis Discovery Protocol : c'est une proposition de description et de découverte de ressources (personne, document, service) via des URL, qui est utilisée dans **OpenID** 2.0 bien qu'externe au projet **OpenID**.
- Provider Authentification Policy Extension (PAPE) : pour le moment toujours en draft, c'est une extension permettant au site (RP) de spécifier au serveur **OpenID** (OP) quelle politique de sécurité doit être appliquée concernant l'authentification de l'utilisateur, et inversement à OP de préciser à RP quelle politique de sécurité a été appliquée lors de l'authentification, afin de renforcer les liens de confiance entre les différents intervenants.

2.3. Mise en place d'un OpenID

2.3.1. OpenID du point de vue de l'utilisateur

- **Utilisation d'OpenID**

Il est vraiment très simple d'utiliser **OpenID**. Tout d'abord, il faut choisir un fournisseur de service **OpenID** (OpenIDFrance, Verisign, MyOpenID, Yahoo, votre serveur personnel) et s'inscrire auprès de ce fournisseur afin d'obtenir une identité numérique **OpenID**, matérialisée par une URL (exemple : http://www.OpenIDfrance.fr /sirpatrick).
Il suffit ensuite de donner cette URL en tant qu'identifiant **OpenID** sur les sites permettant l'authentification via **OpenID**. S'il s'agit de votre première connexion pour cette session,

vous serez redirigé vers le site de votre fournisseur de service **OpenID** pour vous authentifier, puis, si l'authentification est correcte, le site recevra les informations nécessaires à votre connexion s'il ne s'agit pas de votre première connexion pour cette session, le site recevra automatiquement (pour peu que vous ayez autorisé ce site à recevoir des informations vous concernant) les informations nécessaires à votre connexion.
Enfin, vous fermez votre session de navigation, ce qui a pour effet d'effacer les données de votre session d'authentification.

- **Mise en place d'une redirection vers une URL de votre choix**

```
<html>
<head>
<link rel="OpenID.server" href="http://www.OpenIDfrance.fr/index.php" />
< link rel="OpenID.delegate" href="http://www.OpenIDfrance.fr/sirpatrick" />
</head>
<body>
</body>
</html>
```

Les deux lignes insérées entre les balises <head> et </head> permettent de rediriger la requête **OpenID** vers le serveur qui héberge réellement votre identité **OpenID**. Supposons que le code ci-dessus soit placé dans le fichier http://www.drylm.org/jc.lauffer.
Je pourrai alors utiliser cette nouvelle URL à la place de celle originale fournie par OpenIDFrance, tout en conservant mon identité chez eux. Vous bénéficiez ainsi du service du fournisseur d'identité, que vous n'avez pas à gérer et vous avez un identifiant **OpenID** personnalisé, qui reflète peut-être mieux votre identité (http://entreprise/identifiant), mais surtout qui est plus facilement mémorisable ; de plus, si vous changez de fournisseur d'identité **OpenID**, vous changez la ligne de votre fichier HTML et vous continuez à utiliser le même identifiant personnalisé.
Il y a tout de même un inconvénient, chacun jugera de son importance : sans sombrer dans la paranoïa, êtes-vous vraiment sûr de ce que le fournisseur d'identité fait de vos informations personnelles ? Si ce point vous interpelle, sachez que, grâce à **OpenID**, vous pouvez très bien héberger vous-même ces informations. C'est ce que nous allons étudier dans la partie suivante.

- **Mise en place de votre propre serveur d'identité**

Pour mettre un serveur d'identité il nous faut un serveur web avec PHP installé, et sur lequel on peut déposer des fichiers. Ensuite, il nous faut un petit script phpMyID qui permet de mettre rapidement en place un serveur d'identité personnel.
Ce script est composé de plusieurs fichiers. Ceux qui nous intéressent sont :
MyID.php : la bibliothèque implémentant les fonctions de base d'**OpenID** ;
MyID.config.php : le fichier de configuration de notre serveur **OpenID** ;
Il est temps de mettre en place le couple identifiant/mot de passe qui servira à vous authentifier sur votre serveur **OpenID.** Pour cela il faut modifier alors le fichier MyID.config.php avec le login choisi et le hash obtenu : voir l'exemple ce dessous

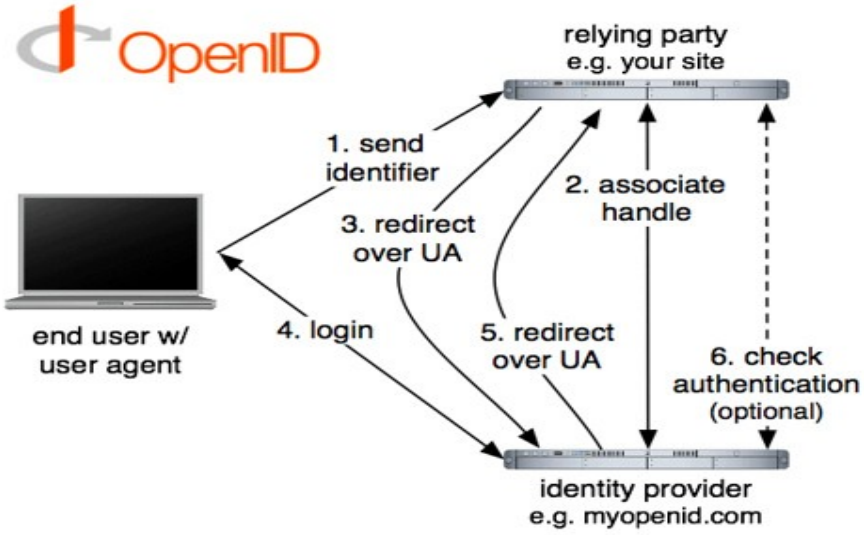
```
/**
* User profile
* @name $profile
* @global array $GLOBALS['profile']
*/
```

```
$GLOBALS[‘profile’] = array( Basic Config – Required ‘auth_username’ => ‘jean’,
‘auth_password’ => ‘58d0a37a9dadec29546e0dce2360dbd7’,[...]);
```

Après, il faut mettez à jour les données SREG (Simple Registration) dans le fichier MyID.config.php :

```
/**
* Simple Registration Extension
* @name $sreg
* @global array $GLOBALS[‘sreg’]
 */
 $GLOBALS[‘sreg’] = array (‘nickname’ => ‘jean-christophe’, ‘email’ =>
‘jean@sirpatrick.com’,‘fullname’ => ‘Jean patrick’,‘dob’ => ‘1979-11-05’,‘gender’ =>
‘M’,‘postcode’ => ‘69100’,‘country’ => ‘FR’,‘language’ => ‘fr’,‘timezone’ =>
‘Europe/Paris’);
```

Le schéma ci montre comme va se dérouler le mécanisme dans ce cas



Enfin, déposez à nouveau le fichier MyID.config.php modifié sur le serveur, puis rechargez la page de configuration pour vérifier la connexion. Après ceci vous possédez une identité numérique personnelle sur un serveur personnel (très simple), ce qui vous garantit que vos informations ne sont pas hébergées (et surtout divulguées) par un tiers.
Mais cette URL n’est pas facile à retenir. Deux choix s’offrent à vous : renommez le fichier MyID.config.php en index.php ou écrivez un fichier index.php contenant ces lignes de code :

```
<html>
<head>
<link rel="OpenID.server" href="http://www.drylm.org/jean-christophe/MyID.config.php" />
<link rel="OpenID.delegate" href=http://www.drylm.org/jean-christophe/MyID.config.php/>
</head>
<body>
</body>
</html>
```

En remplaçant bien évidemment les adresses par les vôtres. Vous pourrez alors utiliser une URL du style http://www.drylm.org/jean-christophe qui est beaucoup plus simple à retenir !

En tant qu'administrateur de site, vous utilisez en général soit une solution toute prête à modifier (Drupal, MediaWiki,...), qui propose en général des extensions pour gérer les différents moyens d'authentification, soit une solution réécrite entièrement, auquel cas il faudra mettre les mains dans le cambouis. Nous allons donc étudier deux exemples de mise en Place pour illustrer ces possibilités.

2.3.2. OpenID du point de vue de l'administrateur de site

- **Mise en place sur Drupal**

Drupal est un CMS, une application Web de gestion de contenu. L'installation du greffon OpenID est des plus simples :
Connectez-vous sur votre site avec un utilisateur disposant des droits d'administration, puis allez dans Administrer ->Modules et activez le module OpenID (disponible dans le paquet de base de Drupal, aucun téléchargement supplémentaire n'est à faire. Rendez-vous ensuite sur la page du profil de votre utilisateur et vous y trouverez un onglet OpenID Identités. Cliquez dessus et ajoutez votre URL d'authentification OpenID. Déconnectez-vous alors de votre session et sur la page principale de votre installation de Drupal, vous pourrez choisir le mode d'authentification OpenID.

- **Mise en place sur votre propre site**

Nous allons construire un petit exemple d'appel à un serveur OpenID pour l'authentification d'un utilisateur en utilisant la bibliothèque php-OpenID. Cet exemple est tiré de l'exemple fourni avec la bibliothèque. L'exemple contient quatre fichiers :
common.php : ce fichier contient les définitions de fonctions globales nécessaires au dialogue avec le serveur OpenID.
index.php : il s'agit du fichier principal, qui sert à saisir l'URL OpenID et à gérer l'affichage des informations par inclusion dans les autres fichiers.
authentification.php : ce fichier contient la première partie du processus d'authentification :
1) création d'un consommateur (qui va servir à dialoguer avec le serveur) ;
2) construction de la requête qui sera envoyée au serveur OpenID sous forme d'URL ;
3) redirection vers le serveur OpenID pour que l'utilisateur s'authentifie auprès de lui.
recuperation.php : ce fichier contient la dernière partie du processus d'authentification :
1) création d'un consommateur (qui va servir à dialoguer avec le serveur) ;
2) récupération de la réponse envoyée par le serveur OpenID ;
3) analyse de la réponse ;
4) affichage des données fournies

3. Sécurité du système d'identifiant unique OpenID

Elle repose sur les liens de confiance qui exister entre les différents acteurs intervenants dans le mécanisme d'authentification. Mais cependant nous savons que Le risque existe pour toutes les technologies d'identification lorsqu'un mot de passe est réutilisable. C'est pourquoi nous allons présenter les risques encourus.

3.1. Le risque de hameçonnage

Une des faiblesses possibles d'**OpenID** est le risque d'hameçonnage, c'est-à-dire de détournement de l'utilisateur vers un autre site que son OP habituel. Ainsi trompé, l'utilisateur donnerait son mot de passe au méchant.

Quelle est l'ampleur du risque ci dessus et comment le limiter?

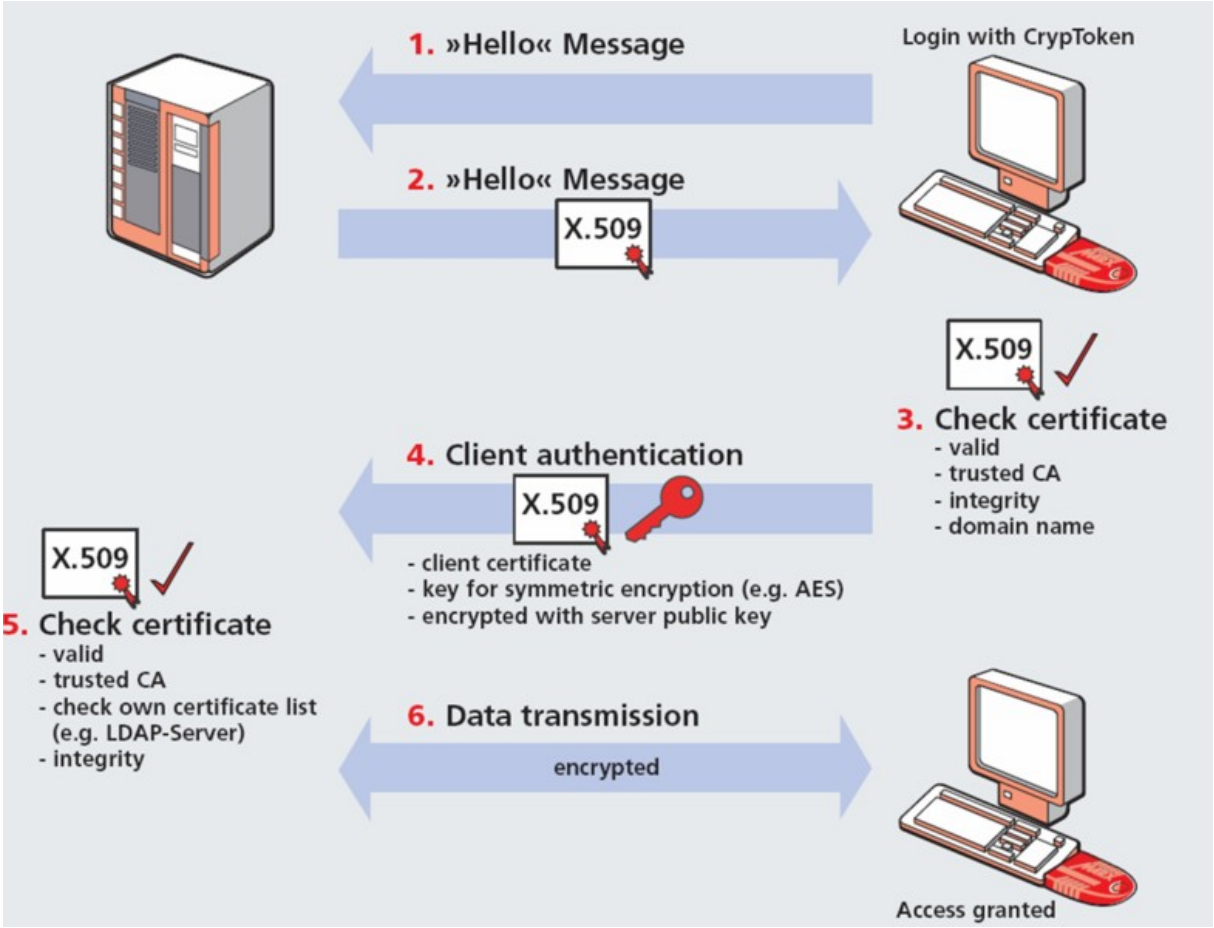
L'expérience montre la vanité qu'il y a à espérer que l'utilisateur scrute le nom de domaine ou le certificat X509, ou sache distinguer le chrome du navigateur de la page Web. Mais **OpenID** a une vulnérabilité particulière, qui est la redirection à laquelle procède le RP. Le RP n'est pas a priori digne de confiance et il pourrait automatiquement rediriger vers un méchant OP. **OpenID**, comme beaucoup de techniques réseaux, fait face ici à la contradiction

entre la facilité d'usage (pour laquelle on a la redirection) et la sécurité.
Le chrome désigne les éléments de l'interface utilisateur qui ne sont pas sous le contrôle du site Web qu'on est en train de regarder, la barre de menus, par exemple. Une technique courante d'hameçonnage est de mettre dans la page Web des éléments rassurants (un cadenas solidement fermé, par exemple), en comptant que l'utilisateur ne voit pas la contradiction avec les éléments du chrome. Comment limiter les risques ? Plusieurs approches sont étudiées. L'une est de supprimer ou de limiter la redirection en obligeant l'utilisateur à taper l'URL de son OP, ou bien en lui faisant utiliser un signet. Cela rendrait l'usage d'OpenID plus difficile. Les techniques anti-hameçonnage classiques peuvent aussi être utilisés. Par exemple, l'OP Peut demander, à la création du compte, que l'utilisateur choisisse une image, qu'il lui montrera à chaque tentative d'authentification. Cette image est un secret partagé entre l'utilisateur et l'OP, un éventuel méchant ne pourrait pas la connaître et donc pas l'afficher. Comme toutes les techniques reposant sur l'éducation de l'utilisateur, on peut craindre que son déploiement soit long et difficile.
Une autre solution serait de rendre l'hameçonnage inutile en supprimant les mots de passe Réutilisables. Si l'OP utilise une authentification à clé cryptographiques asymétriques, un OP Méchant ne pourrait récupérer aucune information utile.

3.2. Risque liée aux certificats des serveurs de TLS (Transport Layer Security)

Avant de commencer, il est nécessaire de savoir comment fonctionne l'authentification par TLS. En outre, TLS diffère de SSL pour la génération des clés symétriques. Cette génération est plus sécurisée dans TLS que dans SSLv3 dans la mesure où aucune étape de l'algorithme ne repose uniquement sur MD5 pour lequel sont apparues quelques faiblesses en cryptanalyse. Par abus de langage, on parle de SSL pour désigner indifféremment SSL ou TLS.
TLS fonctionne suivant un mode client-serveur. Il fournit les objectifs de sécurité suivants:
• l'authentification du serveur ;
• la confidentialité des données échangées (ou session chiffrée) ;
• l'intégrité des données échangées ;
• de manière optionnelle, l'authentification ou l'authentification forte du client avec l'utilisation d'un certificat numérique ;
• la spontanéité, c.-à-d. qu'un client peut se connecter de façon transparente à un serveur Auquel il se connecte pour la première fois ;
• la transparence, qui a contribuée certainement à sa popularité. Du fait que les protocoles de la couche d'application n'ait pas à être modifié pour utiliser une connexion sécurisée par TLS. Par exemple, le protocole HTTP est identique, que l'on se connecte à un schème http ou https. Mais cependant, il est nécessaire de se rendre compte du mécanisme mis en œuvre lors d'authentification du client SSL par certificat numérique.

Dans la majorité des cas, l'utilisateur authentifie le serveur TLS sur lequel il se connecte. Cette authentification est réalisée par l'utilisation d'un certificat numérique X.509 délivré par une autorité de certification (AC). Mais de plus en plus d'applications web utilisent maintenant l'authentification du poste client en exploitant TLS. Il est alors possible d'offrir une authentification mutuelle entre le client et le serveur. Le certificat client peut être stocké en format logiciel sur le poste client ou au format matériel (carte à puce, token USB) pour augmenter la sécurité du lien TLS. Cette solution permet d'offrir des mécanismes d'authentification forte comme le montre le schéma ci-dessous.



Aperçu du fonctionnement de l'authentification client SSL

Dans cette section, il est question de montrer une faille lié au certificat des serveur TLS cette démonstration à été faite par Ben Laurie de l'équipe de garantie appliquée de Google, en collaboration avec un chercheur externe, M. Richard Clayton du laboratoire d'ordinateur, Université de Cambridge, ont pu faire le constat suivant que les divers fournisseurs d'OpenID (OP) ont eu les certificats de serveur de TLS qui ont utilisé des clé faibles c'est-à-dire une clé telle que son utilisation dans une procédure de chiffrement produit un comportement indésirable, en raison du générateur de nombre aléatoire prévisible de Debian.
En combinaison avec la question L'**empoisonnement du cache DNS** et le fait que presque toutes les réalisations de SSL/TLS ne consultent pas CRL, ceci signifie qu'il est impossible à compter sur des ces OP.

Description d'attaque sur ce type de risque

Afin de monter une attaque contre un OP vulnérable, l'attaquant trouve d'abord la clé privée correspondant au faible certificat TLS. Ensuite, il met en place un site web se faisant passer pour OP original à la fois pour le protocole **OpenID** et aussi pour HPPTP/HTTPS.
Puis, L'**empoisonnement du cache DNS** de la victime a faire apparaitre que son serveur est le véritable fournisseur d'**OpenID**.
Il y a deux cas :
On est où la victime est un utilisateur essayant de s identifier, dans ce cas, même si ils utilisent HTTPS qui « s'assurent » que le site qu'ils visitent est en effet leur fournisseur, ils

seront incapable de détecter la substitution et donneront leur procédure de connexion permettant de leur identifier à l'attaquant.
Le deuxième cas est où la victime est la partie composante (RP). Dans ce cas-ci, même si le RP utilise le protocole TLS pour se connecter à l'OP, comme est recommandé pour une plus grande assurance, il ne sera pas protège, car l'immense majorité des implémentations d'**OpenID** ne vérifient pas la CRL (**liste de révocation de certificats)** donc par conséquence accepter les sites malveillants comme les véritables OP.

Mesures d'atténuation pour limiter la vulnérabilité

L'atténuation est étonnamment forte. En théorie, le site vulnérable devrait révoquer leur faible certificat et en délivrer une nouvelle.
Toutefois, depuis la CRL ne va pas certainement être presque cochée, ce qui signifie que le site sera toujours vulnérable aux attaques de la durée de vie du certificat (et peut-être au-delà, en fonction du comportement de l'utilisateur). On remarque que la fermeture du site n'empêche pas l'attaque.
Par conséquent, il est recommandé de mettre en application ces points suivants pour éviter l'atténuation des chutes à d'autres parties.
1. Les navigateurs doivent vérifier CRL par défaut.
2. Les bibliothèques d'**OpenID** doivent contrôler la CRL.
3. Mise en cache DNS résolveurs doivent être patches contre l'attaque par empoisonnement du cache.
4. Soit jusqu'à 1 et 2 ou 3 ont été fait, **OpenID** ne peut pas faire confiance pour une OP qui ne peut pas démontrer qu'il n'a jamais eu un faible certificat.
Toujours est il que des discutions existe sur cette manière de procéder.

Discussion

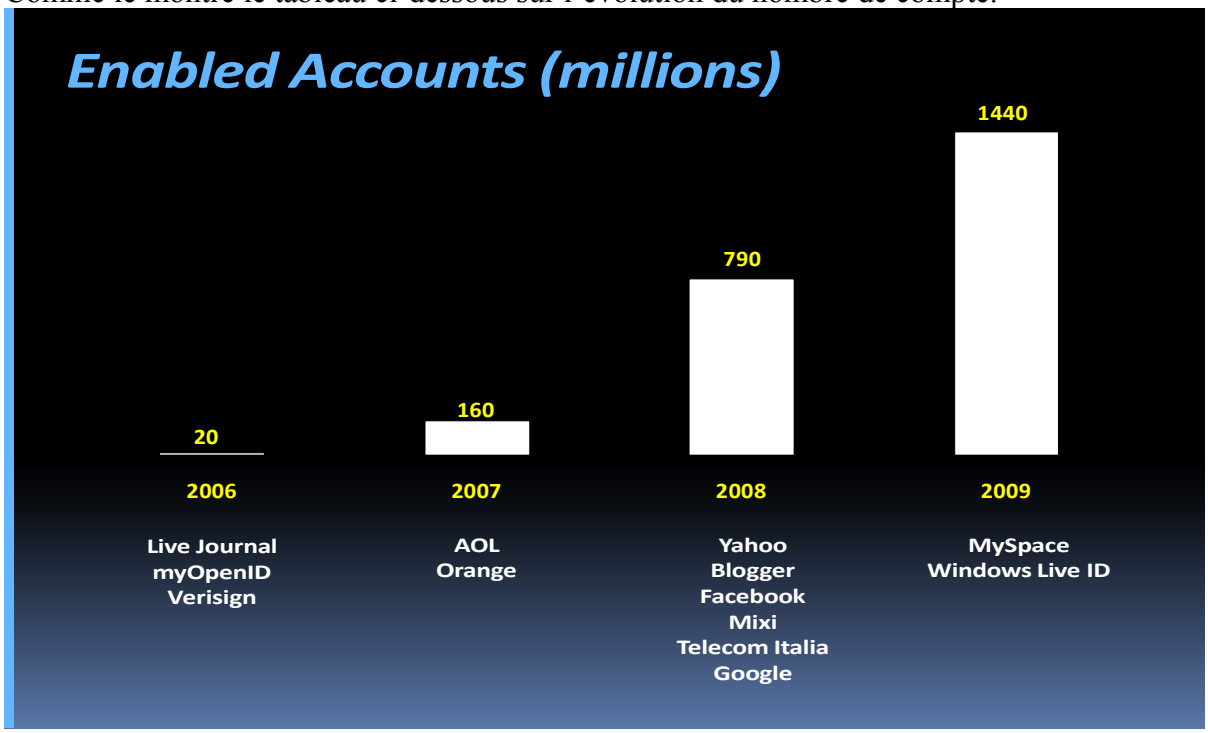
Normalement, quand des problèmes de garantie sont produits avec une d'une seule pièce de logiciel, la chose responsable à faire est à est d'attendre jusqu'à ce que les difficultés soient persistantes avant d'effectuer n'importe quel avis. Cependant, car un certain nombre d'exemples dans le passé ont expliqué, cette approche ne fonctionne pas particulièrement bon quand beaucoup de différents logiciels sont impliqués parce qu'il est nécessaire de combiner une version simultanée des difficultés, tout en espérant que le nombre de gens très grand concernés coopérera à maintenir la vulnérabilité secrète.
Dans la situation actuelle, les difficultés concerneront considérable travaux développement en ajoutant CRL traitant à un grand beaucoup de parties de code d'**openID**. C'est a loin de quantité de travail insignifiante.

Les difficultés concerneront également des modifications aux préférences de programme de lecture pour s'assurer que CRL sont contrôlés par défaut à ce que beaucoup de constructeurs ont résisté pendant des années. Nous sommes extrêmement pessimistes qu'une faille de la sécurité dans **OpenID** sera vue en tant que suffisamment important pour changer les esprits de constructeurs de programme de lecture.
Par conséquent, nous ne voyons aucune valeur en retardant cet avis ; et en effectuant aux groupes le public aussitôt que possible, nous croyons que les personnes qui comptent sur

OpenID seront mieux en mesure de prendre leurs propres différentes mesures pour éviter de compter sur les certificats défectueux que nous avons recensé.
OpenID est au cœur bien un faible protocole, une fois utilisé sous sa forme plus générale, et par conséquent il y a de confiance très limitée sur sa garantie. Ceci signifie que les conséquences de la combinaison des attaques qui sont maintenant possibles n'est rien comme aussi sérieux que la force autrement ont été le cas.
Cependant, il donne une perspicacité dans le type de désastre de garantie qui peut se produire à l'avenir si nous ne commençons pas à prendre CRL les coller sérieusement, mais simplement sur des listes de « to-do » ou les invalider au nom des améliorations des performances minuscules.
Cependant a l'heure actuelle nous ne pouvons par comptabilisé les sites affectés car, il n'y a aucun Bureau d'ordre central des systèmes d'**OpenID**, et ainsi nous ne pouvons pas être sûrs que nous avons recensé tous les faibles certificats qui sont actuellement en service.
Nous pouvons tout simplement affirmer comme Ben et Richard qui ont publié un conseil de garantie à savoir que la vulnérabilité de DNS et l'anomalie de certificat tous deux ont un effet potentiellement dévastateur sur le modèle de garantie d'**OpenID**.

5 Conclusion

OpenID est aujourd'hui un standard stable pour l'identification numérique dans un système décentralisé, largement mise en œuvre dans beaucoup de logiciels (ce qui est un tribut à sa simplicité), mais encore relativement peu déployé.
Comme le montre le tableau ci-dessous sur l'évolution du nombre de compte.



Mais cependant on peut regrette le fait de ne pas bénéficie pas du travail d'une organisation de normalisation, mais il a l'avantage de fonctionner aujourd'hui et de fournir une solution simple aux problèmes des identificateurs multiples, enfermés dans des silos. Mais pose le problème de la question de confiance parce qu'on ne sait réellement a qui on communique,

donc pour ce type de systèmes fonctionne il faut dans un premier temps faire confiance me manière aveugle à une entité donnée. D’où on peut conclure que son avenir est probablement d'être un composant parmi d'autres, des futurs systèmes d'identité décentralisés, pluralistes, et permettant aux utilisateurs de garder ou de reprendre le contrôle de leurs identités.