Chiffrement et hash en PHP contre l'attaque Man in the middle

par Guillaume Affringue (http://guillaume-affringue.developpez.com)

Date de publication : 25/01/2007

Dernière mise à jour : 25/01/2007

Parmi les nombreuses attaques possibles pour voler un compte sur un site internet, l'attaque dite du "Man in the middle" est probablement une des plus faciles et efficaces.
Nous allons voir ici plusieurs méthodes de chiffrement et de hashage des données pour s'en protéger.

I - Introduction
    I-A - Remerciements
    I-B - Prérequis
    I-C - Préambule
II - Origine de ces attaques
    II-A - Présentation de l'attaque « Man in the middle »
    II-B - Les vols
        II-B-1 - Session
        II-B-2 - Mot de passe
        II-B-3 - Cookie
III - Protection contre le vol de mot de passe
    III-A - Hash du mot de passe
        III-A-1 - Utilité du hash
        III-A-2 - Mise en place
            III-A-2-a - Coté serveur
            III-A-2-b - Coté client
    III-B - Technique du Grain De Sel (GDS)
        III-B-1 - Utilité du GDS
        III-B-2 - Présentation des 3 techniques possibles
            III-B-2-a - GDS global
            III-B-2-b - GDS par utilisateur
            III-B-2-c - GDS par session
        III-B-3 - Mise en place du GDS par utilisateur
IV - Protection contre le vol de session
    IV-A - Identification par l'environnement
        IV-A-1 - L'adresse IP
        IV-A-2 - Variable d'environnement
        IV-A-3 - Conclusion
    IV-B - Système de ticket
V - Chiffrement des données
    V-A - Algorithmes existants
        V-A-1 - Chiffrement symétrique
        V-A-2 - Chiffrement asymétrique
    V-B - Cas pratique avec le RSA
VI - Chiffrement de la session : HTTPS
    VI-A - Présentation
    VI-B - Déploiement
    VI-C - Certificat
VII - Conclusion
VIII - Annexes
    VIII-A - Modifications du package Crypt_RSA de PEAR
IX - Sources
X - Liens

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

I - Introduction

I-A - Remerciements

Je remercie Yogui, pour son soutien depuis que je suis rentré dans la rédaction de developpez et pour ses corrections.

Je remercie toutes les personnes qui m'ont aidés à peaufiner cet article, et ceux qui m'ont encouragés après la lecture (ça fait toujours plaisir).

Je remercie toutes les personnes qui interviennent de près ou de loin à l'élaboration de ces articles liés à la sécurité.

I-B - Prérequis

Théorique

Ce cours s'adresse aux débutants ayant malgré tout un savoir suffisant en PHP pour exploiter correctement les sessions.

Les codes sources proposés ici ne sont pas vraiment compliqués, mais une bonne connaissance du web et du rôle de chaque langage (notamment PHP et JS) est nécessaire pour les déployer.

Software

Tous les exemples/sources présentés ici ont été réalisés sous Wamp 1.6.6, utilisant PHP 5.2.0.

Il faudra bien évidemment un navigateur internet avec Javascript activé. Pour afficher mes exemples, j'ai utilisé Firefox 2.0.0.1.

Pour voir le résultat des attaques et les solutions, nous allons utiliser Ethereal, qui est devenu depuis peu Wireshark. Ce logiciel écoute le réseau et capture absolument tout.

Je montrerai ici plusieurs captures d'écrans qui reflètent les propos énoncés, je vous invite à vérifier vous même la facilité d'execution de la plupart des tests que je réaliserai.

I-C - Préambule

Les techniques décrites ici protègent de l'attaque du "Man in the middle". Elles ne sont pas spécifiques à PHP et Javascript et sont utilisables avec tous les langages Web.

Elles sont fondées sur l'utilisation de techniques de chiffrement et de hash.

Pour commencer, vous devons bien comprendre la différence entre chiffrement et hashage. Pour cela, vous pouvez vous référer au dictionnaire : **Hashage** et **Chiffrement**

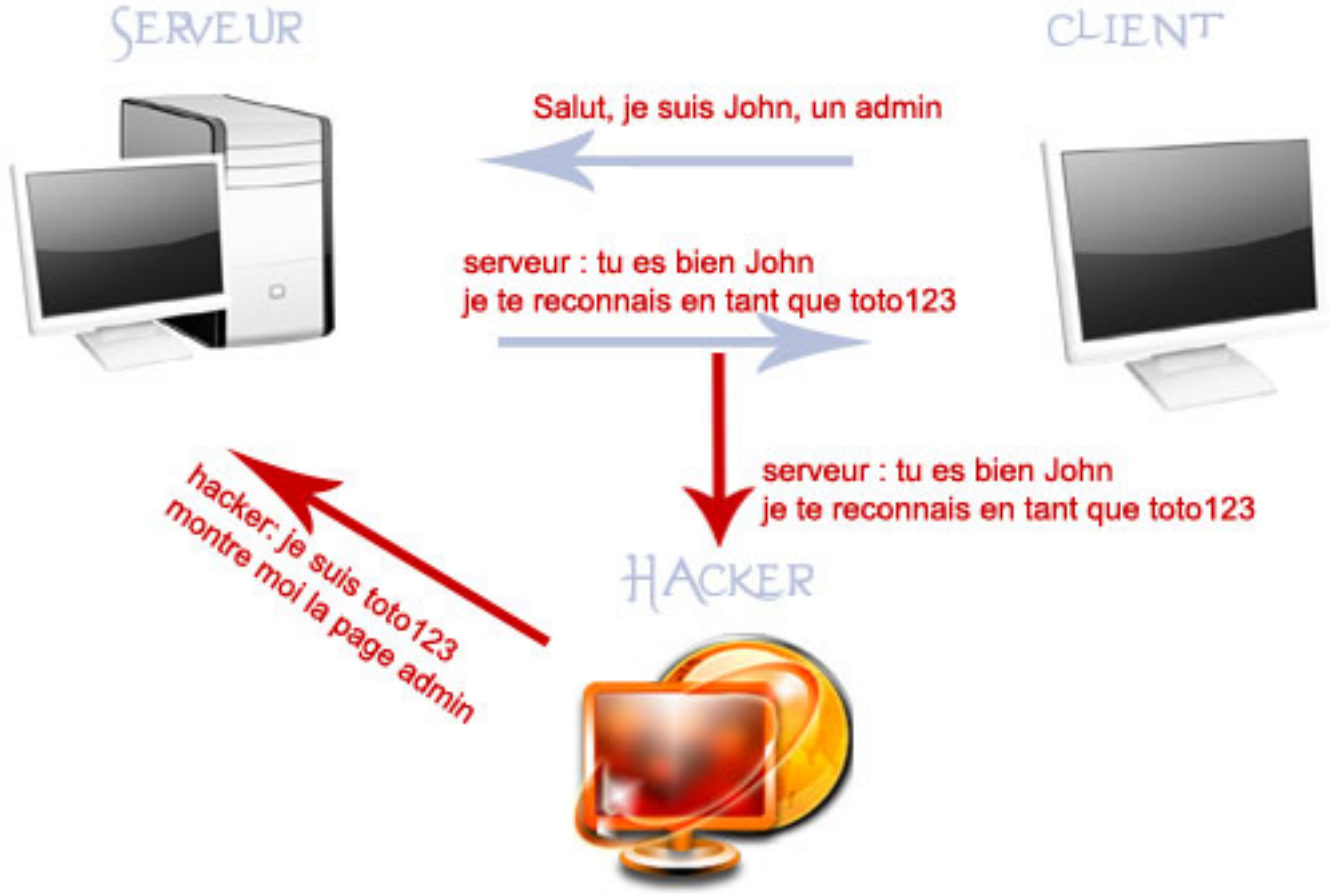
Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

II - Origine de ces attaques

II-A - Présentation de l'attaque « Man in the middle »

L'attaque « Man in the middle » consiste pour le pirate à se brancher sur la ligne entre vous et le serveur, à écouter ce qui transite, puis, au bon moment, à essayer de prendre votre place.

Dans l'exemple suivant, John est le pseudonyme d'un administrateur se connectant au site, toto123 est l'identifiant de session.



Schématisation d'une attaque Man in the middle

Cette technique est très facile à mettre en place dans les entreprises ou les universités, car il y a souvent une seule passerelle entre internet et le reste du réseau. Cela signifie que toute donnée envoyée sur internet par un ordinateur du réseau sera susceptible d'être « entendue ». Le pire scénario se produit lorsque le matériel est un peu vieux ou que l'installation a été réalisée à moindre coût et que les boîtiers permettant la mise en réseau sont des hubs et non des switchs. Le hub, lorsqu'il reçoit un message provenant d'un ordinateur, se contente de l'amplifier et de l'envoyer à tout le monde, alors que le switch n'envoie qu'à la bonne machine. Ce qui veut dire qu'avec un logiciel comme Ethereal sur un réseau constitué de hubs (il y en a encore beaucoup), un hacker pourrait prendre la place de n'importe quel membre sur n'importe quel site ouvert depuis le réseau.

II-B - Les vols

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

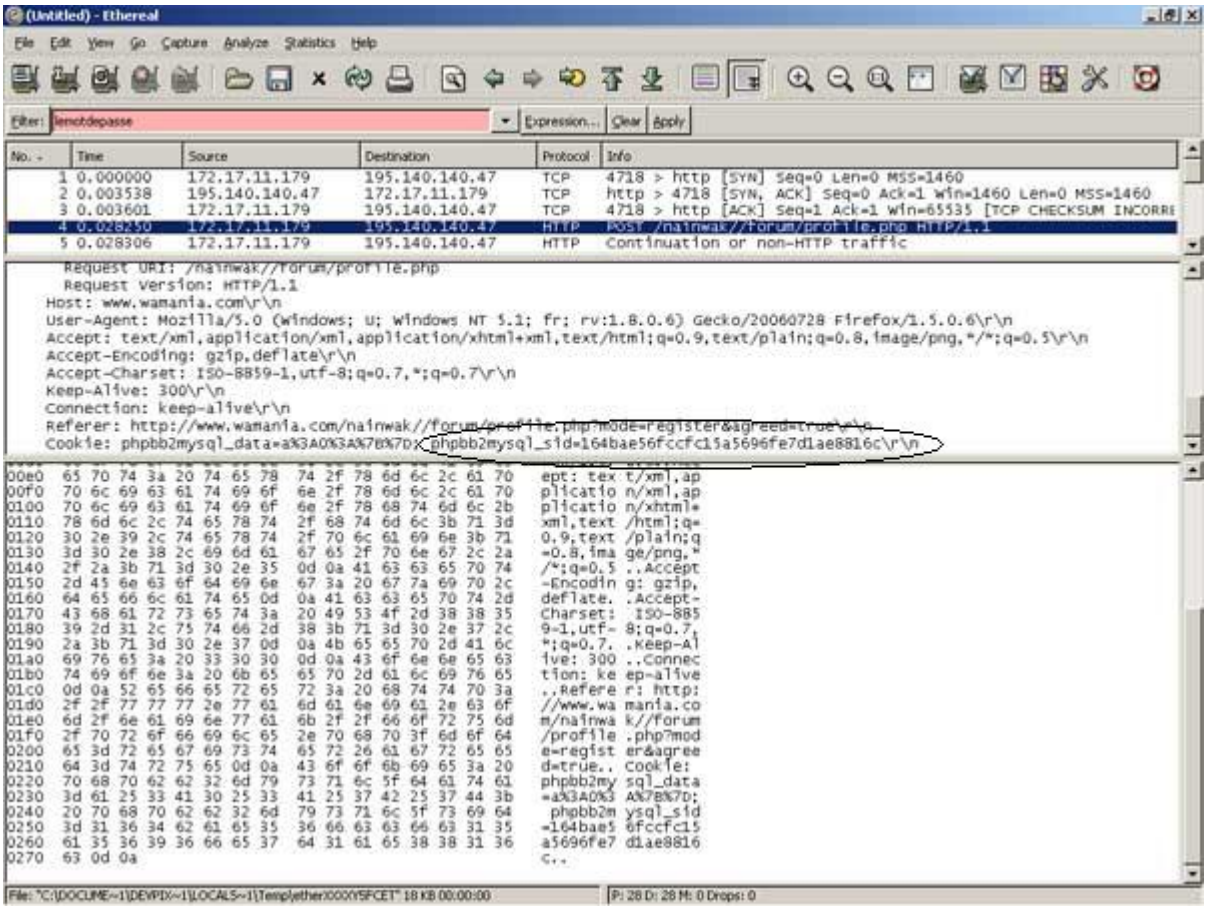
Nous avons vu comment il est possible pour un hacker d'écouter le réseau. Mais que peut-il voler exactement ?

II-B-1 - Session

Pour voler une session, il suffit de récupérer l'identifiant de session. Cet identifiant est unique et propre à une et une seule session à un instant donné. L'identifiant de session en PHP peut être transmit par deux moyens, cookie ou url.

Le passage par URL est une faille de sécurité. Imaginons ce cas de figure : Vous naviguez sur un site qui transmet l'identifiant de session par URL, par exemple un forum comme phpBB. Un visiteur (mal intentionné) que nous nommerons Mr X, poste un message avec un lien sur son site perso, avec en général beaucoup de pipo pour attirer les gens. Sur ce site, Mr X a écrit un script qui récupère la variable globale $_SERVER['HTTP_REFERER'] qui contient la dernière adresse visitée, c'est-à-dire celle du forum avec l'identifiant de session dedans. Le script de Mr X va repérer cet identifiant et Mr X pourra alors l'utiliser pour voler la session.

Que ce soit par URL ou par cookie, l'attaque du "Man in the middle" permet de récupérer l'identifiant de session. Voici ici une capture d'écran du logiciel Ethereal. Cette capture d'écran présente ce que le logiciel a enregistré dans le court intervalle de temps qu'il m'a fallu pour valider une inscription sur un forum PHPBB. On voit ici l'entête HTTP correspondant à l'envoi du cookie contenant l'identifiant de session. Le vol de cet identifiant est d'autant plus facile que le cookie est envoyé avec chaque élément de la page, images comprises.

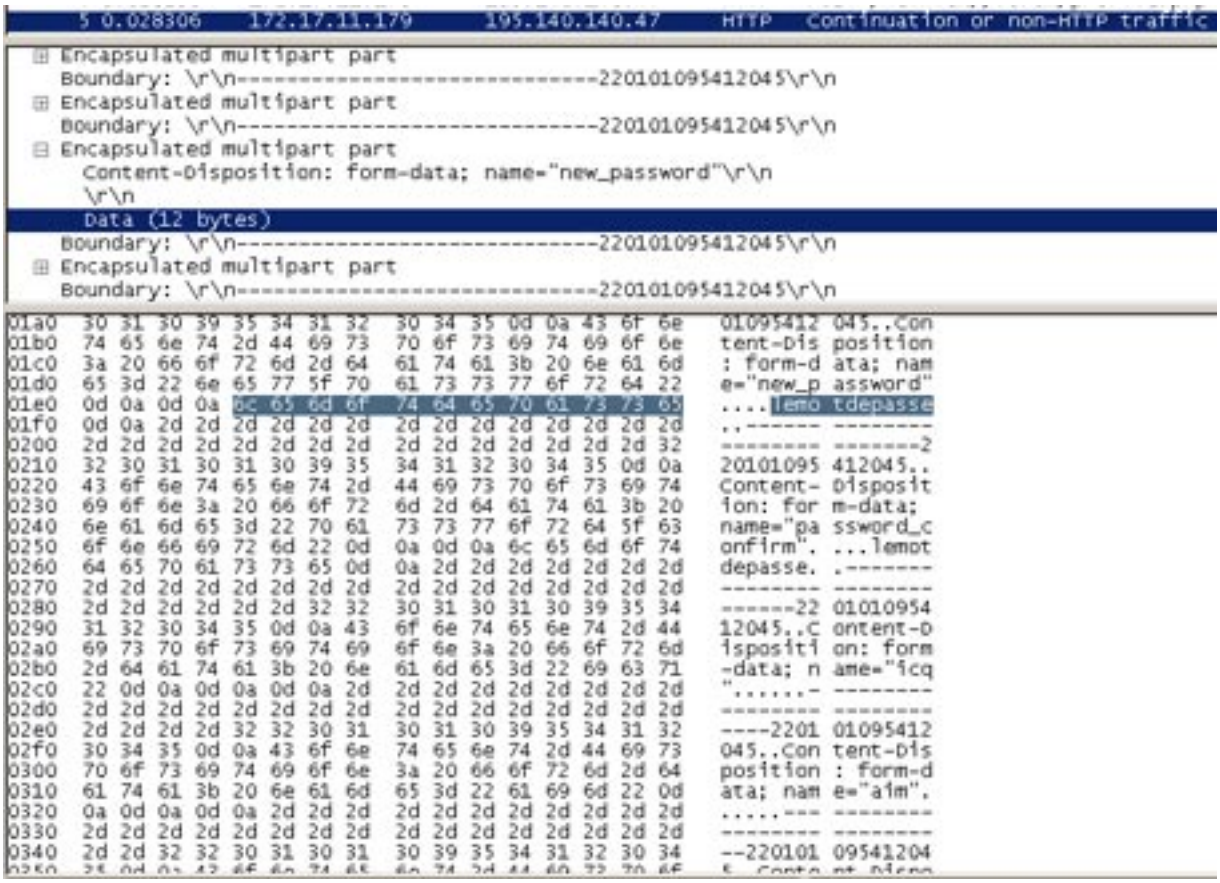


Identifiant de session capturé grâce à Ethereal

II-B-2 - Mot de passe

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Le pire scénario, le vol du mot de passe. Ce vol est aussi le plus dur, car le hacker n'a souvent qu'un seul essai, lors de la phase d'identification (ou d'inscription). Dans un espace membre classique, autant lors de l'inscription que de la connexion, le mot de passe circule en clair.

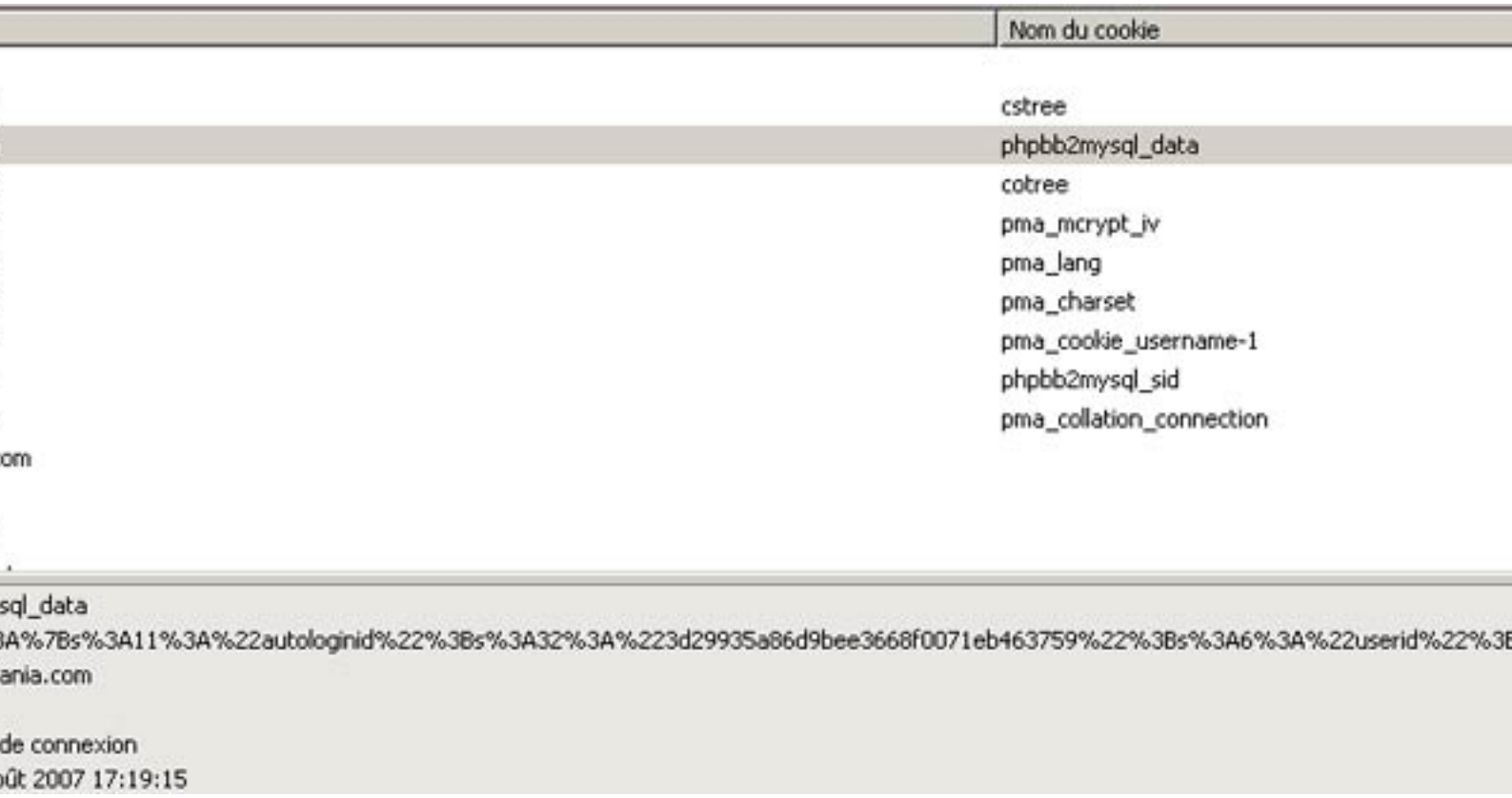


Mot de passe capturé grâce à Ethereal

II-B-3 - Cookie

Il existe 2 types de cookie permettant l'identification : le cookie de session, tel qu'on l'a vu précédemment, ne contenant en général que l'identifiant de session, et un cookie dit « d'autologin » qui permet à un visiteur de ne pas saisir de nouveau son login et son mot de passe lors d'une visite ultérieure. Le vol peut, comme on l'a vu, se faire par l'attaque du "Man in the middle", mais peut aussi se faire par un vol physique (pour les cookies d'autologin en général). Une personne utilise votre PC, fouille dans vos cookies et copie par exemple sur une clé USB les cookies d'autologin qui l'intéressent. Elle les copie chez elle, et se connecte ensuite directement sur votre compte sur le site concerné. La seule solution pour éviter ceci est de ne pas faire de cookie d'autologin et de forcer le visiteur à se reconnecter manuellement à chaque fois.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.



Affichage du contenu d'un cookie sur Firefox

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

III - Protection contre le vol de mot de passe

Pour éviter le vol du mot de passe, il faut limiter, voire proscrire, sa transmission et son stockage en clair. À la place, on va manipuler un hash, beaucoup sécurisé car il est (théoriquement) impossible à inverser. En pratique, c'est "réversible". Le hacker utilisera par exemple des dictionnaires pour obtenir le mot de passe en clair. Nous verrons comment lui compliquer la vie avec la technique du grain de sel.

III-A - Hash du mot de passe

III-A-1 - Utilité du hash

Le hash est un procédé de chiffrement irréversible. Ceci implique qu'on ne peut appliquer la transformation inverse, et qu'on ne peut donc que comparer les hashs entre eux. L'avantage est que personne, y compris le webmaster, n'a accès aux mots de passe des membres.

Il existe deux principaux algorithmes de hash, le MD5 et le SHA1.

Le MD5 est déprécié et NE DOIT PLUS ETRE UTILISE. En effet, une faille dans l'algorithme a été trouvée, permettant de diminuer considérablement le nombre d'opérations nécessaires à un inversement par rapport à une attaque force brute. De plus, et c'est le plus inquiétant, il a été montré qu'il était possible de créer volontairement des collisions avec un document donné. En effet, les hashs MD5 sont constitués de 32 caractères alphanumériques, soit $36^{32} = 6.33^{49}$ combinaisons possibles. Ainsi, le nombre de hashs possibles est limité. Or, nous réalisons des hashs de chaîne de caractères (strings, fichiers...) de taille indeterminée et pour la plupart bien plus grandes que 32 caractères. Par exemple, nous pouvons constituer 2^{1556} chaînes de 1000 caractères alphanumériques.

Nous avons donc une infinité de chaîne de caractères pouvant donner un hash précis et il est maintenant possible de déterminer instantanément les autres chaînes de caractères à partir d'une des chaînes et du hash.

Il est donc recommandé d'utiliser des algorithmes plus robustes, comme le SHA1 ou le SHA-256. Mais il faut bien garder en tête qu'un algorithme est reconnu comme fiable uniquement car il n'a **pas encore** été craqué. Ainsi, le SHA1 est déjà sur la pente descendante, le nombre d'opérations nécessaires à l'inversion ayant été ramené de 2^80 à 2^63.

Maintenant que nous avons présenté le principe du hash et les principaux algorithmes, nous allons voir un exemple pratique de mise en place côté serveur et côté client.

Vous pouvez tester un code de ce type sur votre serveur pour avoir la liste des hashs supportés par votre installation de PHP.

Afficher la liste des hashs supportés

```php
print_r(hash_algos());
```

Résultat

```
Array
(
    [0] => md4
    [1] => md5
    [2] => sha1
    [3] => sha256
```


Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Résultat
```
        [4] => sha384
        [5] => sha512
        [6] => ripemd128
        [7] => ripemd160
        ...
    );
```

III-A-2 - Mise en place

III-A-2-a - Coté serveur

Le hash coté serveur empêche, en cas de vol de la base de donnée par intrusion sur le serveur, de fournir au hacker tous les mots de passe en clair des membres du site. Il faut savoir que la plupart des membres ne possèdent que 2 ou 3 mots de passe qu'ils utilisent sur presque tous les sites. Le webmaster doit donc s'assurer de son côté de l'intégrité des mots de passe de ses visiteurs. Pour effectuer un hash coté serveur, il existe la fonction SHA1 incorporée à PHP. On l'appelle lors de l'inscription du membre puis, à chaque connexion, on compare les 2 hashs.

Manipulation du hash lors de l'inscription
```php
// A l'inscription
$pass = $_POST['pass'];
$login = $_POST['login'];
mysql_query("
    INSERT INTO membres (login, pass)
    VALUES ('".mysql_real_escape_string($login)."', '".sha1($pass)."')");
```

Manipulation du hash à la connexion
```php
// A la connexion
$pass = $_POST['pass'];
$login = $_POST['login'];
$result = mysql_query("
    SELECT 1
    FROM membres
    WHERE (login='".mysql_real_escape_string($login)."')
    AND (pass='".sha1($pass)."')");
if (mysql_num_rows($result) > 0)
{
    // on connecte
}
```

III-A-2-b - Coté client

L'utilisation du hash coté client garanti l'intégrité du mot de passe lors de son transit depuis le navigateur du client vers le serveur. Il faut donc encoder le mot de passe **avant** de l'envoyer. Ceci peut être fait par un langage de script comme le Javascript. La mise en place de cette technique est plus lourde que le hash coté serveur (car ces langages de script n'implémentent pas de fonction MD5 ou SHA1), et peut ne pas fonctionner si le client a désactivé Javascript. Il faudra alors prévoir une méthode alternative (transmission du mot de passe en clair). La première étape sera donc d'écrire, ou de récupérer une fonction de hash pour le langage de script que vous aurez choisi. Nous utiliserons ici une implémentation en Javascript de l'algorithme SHA1.

Réaliser un hash coté client n'empêche absolument pas le vol de session, il suffira au hacker d'envoyer le hash au lieu du mot de passe. Il faut bien comprendre que l'utilisation du hash n'est faite que pour protéger le mot de passe.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

*Plusieurs sites réputés utilisent cette technique. Par exemple, si vous fouillez un peu sur la page d'identification de Yahoo!, vous trouverez un lien vers **ce fichier**. Cette même implémentation est utilisée dans d'autres scripts, notamment de forum.*

Le fichier Javascript que nous utiliserons pour le sha1 provient de la même source dont le lien est disponible en fin d'article.

Un fichier opérationnel est disponible ici (**sha1hash.zip**)

Formulaire coté client

```
<script type="text/javascript" src="sha1hash.js"></script>
<form method="post" action="" id="connexion-form" name="connexion-form"
 onsubmit="sha1hash(this.mdp, this.sha1mdp);">
 Pseudo : <input type="text" name="login" id="login" size="20" autocomplete="off">
 Mot de passe : <input type="password" name="mdp" size="20" />
 <input type="hidden" name="sha1mdp" value="" />
 <input value="Valider" id="Valider" type="submit" class="bouton">
</form>
```

- L'utilisateur rempli les 2 champs textes login et mdp.
- Lorsqu'il envoie le formulaire, la fonction sha1hash est appelée. Cette fonction prend en argument les objets correspondant aux champs de formulaire mdp et sha1mdp.
- La fonction remplit le champ sha1mdp avec le mot de passe hashé et **vide le champ correspondant au mot de passe en clair.**
- Le formulaire est envoyé au serveur.

Recéption du formulaire coté serveur

```
// Javascript activé
if ( ! empty ($_POST['sha1mdp']) )
{
 $sha1mdp = $_POST['sha1mdp'];
}
// Javascript désactivé
elseif ( ! empty($_POST['mdp']) )
{
 $sha1mdp = sha1($_POST['mdp']);
}
```

III-B - Technique du Grain De Sel (GDS)

III-B-1 - Utilité du GDS

Le but premier du GDS est de protéger le hash du mot de passe lors de son transfert et de son stockage. En effet, il est de plus en plus facile d'inverser un hash, nous utilisons donc le GDS afin de compliquer cette inversion.

Le grain de sel a 2 intérêts très distincts suivant s'il est utilisé côté client ou serveur.

Côté serveur

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Cela signifie en pratique que l'on a hashé le mot de passe de l'utilisateur, préalablement concaténé avec le GDS, avant de l'enregistrer dans la base de données.

Ceci empêche le hacker, en cas de vol de la base de données, d'utiliser la force brute ou un dictionnaire pour inverser les hashs. En effet, comme le GDS est secret, le hacker ne pourra pas constituer de dictionnaire adapté. De plus, la force brute sera vraiment plus longue, car si le GDS est une chaîne de 32 caractères et le mot de passe une chaîne de 8 caractères alors il faudra tester 2.8^{12} combinaisons sans GDS ou 1.8^{62} avec GDS.

Le fondement de cette protection est que le hacker ne connaît pas la valeur du GDS.

Cette utilisation du GDS n'empêche absolument pas la force brute provenant d'un formulaire utilisateur. Pour empêcher cela, il faudra mettre un timer entre chaque tentative ou un bloqueur au bout de x tentatives.

Côté client

Le GDS côté client suppose que le GDS ne soit pas secret. Il ne peut donc pas protéger des attaques par force brute. Par exemple, le mot de passe est "lol" et le GDS "azerty". On appliquera le hash sur toutes les combinaisons possibles de "aaa" à "zzz" pour comparer chaque résultat au hash qu'on souhaite inverser. Avec un GDS connu, il suffira de tester les combinaisons de "azertyaaa" à "azertyzzz". Cela ne complique en rien notre recherche.

Ce type de GDS protège contre les attaques par dictionnaire. En effet, depuis quelques temps, des dictionnaires de hash sont disponibles sur le net, voire directement utilisables en ligne. On peut trouver des dictionnaires en ligne contenant plus de 500 millions de hashs référencés. Grâce à ces dictionnaires, il est possible de retrouver instantanément un mot de passe à partir du hash **s'il est dans le dictionnaire.**

Toute l'utilité de ce GDS est contenu dans ces mots "s'il est dans le dictionnaire", car si notre mot de passe "lol" y est très certainement, il y a peu de chances que "azertylol" y soit, et encore moins "skndgqzmsrgze6r4gb5q1fde5bhs3df4b3q4df3b4sdfb**lol**".

Avoir un site utilisant une technique de GDS ne change rien au fait que le choix du mot de passe est crucial pour la sécurité. Il est donc conseillé d'utiliser des mots hors dico avec chiffres / majuscules / minuscules / caractère spéciaux.

	Protège		Contre	
	Base de données	Transfert du mdp	Attaques par force brute	Attaques par Dictionnaire
Côté server	X		X	X
Côté client		X		X

On voit dans ce tableau récapitulatif la faiblesse du "GDS côté client". Si le hacker arrive à recueillir le GDS et le hash côté client, il pourra utiliser une technique de force brute traditionnelle, sans que nous lui ayons compliqué le travail. Le GDS côté serveur et côté client sont deux techniques complémentaires.

III-B-2 - Présentation des 3 techniques possibles

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Il existe en réalité beaucoup plus de méthodes, mais elles impliquent toutes -à au moins l'une des étapes- le stockage ou la transmission du mot de passe en clair.

Nous noterons ici pour chaque technique l'efficacité de la protection correspondante :

- 0 = pas protégé
- 1 = peu protégé
- 2 = bien protégé
- 3 = très bien protégé

III-B-2-a - GDS global

Le principe ici est d'avoir un GDS général pour tout le site et pour tous les utilisateurs. L'avantage est la facilité de mise en oeuvre ; cependant, c'est également la moins sûre. En pratique, on a un GDS donné. À l'inscription on enregistre le mot de passe en base concaténé avec ce GDS et hashé en SHA1. Lorsqu'un visiteur se loggue, on envoie le GDS au client, on hash en SHA1 le mot de passe concaténé avec ce GDS, on envoie au serveur. Celui-ci compare simplement les deux valeurs.

Inconvénients

Il suffit de constituer un seul dictionnaire pour attaquer tous les membres du site. De plus, le GDS n'est pas secret, donc le temps des attaques par force brute n'est pas augmenté.

Avantages

La technique est simple, la base est protégée contre les attaques par dictionnaire et mot de passe ne circule pas en clair.

	Protection contre force brute	Protection contre dictionnaires
Côté serveur	1	2
Côté client	0	2

III-B-2-b - GDS par utilisateur

C'est la technique que j'ai adoptée. Elle est similaire à la précédente, sauf qu'à l'inscription, on génère un GDS qui sera propre à l'utilisateur. Ce GDS sera conservé dans la base et envoyé au client à chaque demande de connexion.

Mise en place

- À l'inscription, on génère le GDS et on le place en base
- À la connexion, le visiteur entre son login
- On va chercher le GDS, soit par Ajax, soit en envoyant le login par formulaire, et en renvoyant le GDS
- On demande le mot de passe, on chiffre coté client et on envoie au serveur

Inconvénients

La récupération du GDS après la saisie du login implique soit deux formulaires (comme dans SPIP), soit un script Ajax

Avantages

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Tout est protégé contre les attaques par dictionnaire et le hacker doit constituer un dictionnaire par membre, ce qui est fastidieux et décourageant.

Le GDS de l'utilisateur est enregistré dans la base de données et transite à chaque connexion. Le hacker y a donc accès facilement et peut donc utiliser la force brute pour inverser le hash.

	Protection contre force brute	Protection contre dictionnaires
Côté serveur	0	3
Côté client	0	3

III-B-2-c - GDS par session

Cette technique consiste à générer un GDS propre à la session. On ne pourra donc pas stocker le mot de passe en base grâce à ce GDS. Il faudra donc prévoir une deuxième technique (GDS global ou par utilisateur) pour protéger les mot de passes enregistrés.

Inconvénients

Difficile à mettre en oeuvre, car elle nécessite une deuxième technique pour protéger le stockage en base car on ne peut pas enregistrer en base un mot de passe protégé par un GDS qui changera. Il faudrait gérer 2 GDS, et effectuer côté client l'opération lourde

```
protected_mdp = SHA1(SHA1(mdp+GDS1)+GDS2)
```

Dans laquelle SHA1(mdp+GDS1) représente le mot de passe tel qu'il est stocké en base.

Ainsi, toute la base est protégée par le même GDS, et en outrepassant le GDS de session côté client (rendu possible car on doit prendre en compte la possibilité que Javascript ne soit pas activé), on peut réaliser une attaque par dictionnaire dans lequel le dictionnaire serait constitué pour ce GDS spécifique.

On revient au final au principe du GDS global.

Avantages

Probablement la technique la plus sûre si on ne prend pas en compte la possibilité d'avoir le Javascript désactivé, car même si le hacker réussit à trouver le GDS, celui-ci changera pour chaque session (et ainsi pour chaque membre).

	Protection contre force brute	Protection contre dictionnaires
Côté serveur	1	2
Côté client	0	3

III-B-3 - Mise en place du GDS par utilisateur

Nous allons ici approfondir la technique de protection par GDS propre à un utilisateur car c'est la seule protégeant réellement d'une attaque par dictionnaire, étant donné qu'il faut constituer un dictionnaire par membre. Pour la réaliser, je vais utiliser un peu d'Ajax. Comme je l'ai dit plus haut, il est possible de réaliser 2 formulaires au lieu du script Ajax, c'est le choix qu'a fait le projet SPIP (cf. rubrique lien à la fin de l'article). Le script Ajax utilisé s'appuie sur le tutoriel de Denis Cabasson disponible ici **Tutoriel Ajax**

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Le principe consiste à lire le champ de login et à appeler une page distante à chaque changement observé. Ainsi, à chaque saisie d'un caractère dans le champ login, nous vérifierons si un utilisateur existe avec ce login et si oui, nous renverrons son GDS.

Un fichier JS pour réaliser ceci est disponible ici (**getGDS.zip**).

Il nous manque maintenant le formulaire et un fichier PHP renvoyant le GDS d'un utilisateur.

Côté client

Il est nécessaire dans la page d'insérer et d'initialiser les scripts de GDS et de hash, puis nous affichons le formulaire.

Insertion du code Javascript dans l'entête de l'HTML

```
<script type="text/javascript">
 var _adresseRecherche = "gds.php"   // l'adresse à interroger pour trouver le GDS
</script>
<script type="text/javascript" src="sha1hash.js"></script>
<script type="text/javascript" src="getGDS.js"></script>
<script type="text/javascript">
 window.onload = function()
 {
  initGetGDS(
   document.getElementById('connexion-form'),
   document.getElementById('login'),
   document.getElementById('Valider'),
   document.getElementById('gds'),
   document.getElementById('info_gds')
   );
 };
</script>
```

La variable JS _adresseRecherche définit la page distante qu'il faudra appeler pour récupérer le GDS.

Pour initialiser le listener du champ login, nous appelons la fonction initGetGDS en lui passant comme paramètres les objets DOM

- Du formulaire
- Du champ login
- Du bouton de submit
- Du champ hidden dans lequel placer le GDS
- Du div dans lequel on pourra indiquer si le GDS est bien réceptionné. (optionnel)

Code HTML du formulaire

```
<form method="post" action="" id="connexion-form" name="connexion-form"
 onsubmit="sha1hash(this.mdp, this.sha1mdp, this.gds);">
 Pseudo : <input type="text" name="login" id="login" size="20" autocomplete="off" />
 <div id="info_gds">Grain de sel</div><br />
 Mot de passe : <input type="password" name="mdp" size="20" /><br />
 <input type="hidden" name="sha1mdp" value="">
 <input type="hidden" name="gds" id="gds" value="">
 <input value="Valider" id="Valider" type="submit" class="bouton">
</form>
```


Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Le reste est identique au modèle présenté dans le chapitre consacré au hash, avec le GDS en plus. Le hash se fera ici selon la formulaire :

sha1mdp = sha1(GDS + mdp + GDS)

Côté serveur

Il s'agit d'un script très simple qui recherche dans la base de données ce login, sélectionne le GDS et en fait un echo. On pourrait établir un flux XML plus propre, mais pour l'exemple, un simple flux texte est suffisant.

Code php du fichier gds.php

```php
if(isset($_GET['login']))
{
   $login = urldecode($_GET['login']);
}
else
{
   exit 0;
}

$gds = $db->get_var("SELECT me_gds FROM membres WHERE
 login='".mysql_real_escape_string($login)."'");
if ( !empty($gds) )
{
    echo $gds;
}
```

Le login est envoyé par GET, on le récupère, on applique un urldecode car on a utilisé urlencode coté Javascript (dans le fichier getGDS.js).

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

IV - Protection contre le vol de session

IV-A - Identification par l'environnement

IV-A-1 - L'adresse IP

Pour éviter le vol de session, l'enregistrement de l'adresse IP apparaît comme une solution évidente. En effet, une machine -> une session -> un utilisateur. Si l'IP change, cela signifie que la session a été volée et utilisée sur une autre machine. Pour réaliser ce contrôle, c'est très simple, une fonction de ce type retourne l'IP de l'utilisateur.

Fonction PHP récupérant l'IP de l'utilisateur

```php
/**
     * @desc Récupére l'adresse IP du client
     * @return string
     * @access public
     */
    public function getIP()
    {
        if(getenv('HTTP_X_FORWARDED_FOR'))
        {
            return getenv('HTTP_X_FORWARDED_FOR');
        }
        elseif(getenv('HTTP_CLIENT_IP'))
        {
            return getenv('HTTP_CLIENT_IP');
        }
        else
        {
            return getenv('REMOTE_ADDR');
        }
    }
```

En théorie, c'est parfait.

En pratique, cette méthode est pour moi inutile et même parfois embêtante, d'une part l'IP n'est pas une valeur sûre, d'autre part l'IP d'un même utilisateur peut changer au cours du temps.

Session Hijacking et IP spoofing

Je ne m'attarderai pas trop sur ces notions, il y a juste besoin de savoir qu'une IP, ça se vole. Pour en savoir plus, vous pouvez aller voir les articles de David Burgermeister et Jonathan Krier **(ici)**.

IP instable

Certains fournisseur comme AOL et certaines entreprises ou universités renouvellent très souvent l'IP de leurs ordinateurs, rendant impossible l'existence d'une session se reposant dessus.

Fonder son authentification sur l'IP revient à renoncer à beaucoup d'utilisateurs potentiels.

IV-A-2 - Variable d'environnement

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Une autre méthode pour avoir une "signature" d'un visiteur est d'utiliser son environnement, c'est-à-dire son explorateur, son système d'exploitation, *etc*.

On est sûr que ces données ne changeront pas pendant la session.

Par contre, comme l'IP, c'est loin d'être involable. En fait, c'est même inutile car si le hacker a déjà volé l'identifiant de session par l'écoute du réseau, il aura aussi capturé ceci :

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Capture des entêtes HTTP lors de l'appel d'une page

IV-A-3 - Conclusion

En conclusion, se fier à des données liées à un utilisateur pour fiabiliser la session n'est pas une bonne idée, car ces données proviennent justement de l'utilisateur ;-) et la première règle en développement d'application web est de ne jamais faire confiance à ce qui vient de l'extérieur, adresse IP comprise.

Il faut évidemment relativiser, le vol d'adresse IP n'est pas donné à tout le monde, mais je déconseille malgré tout cette technique à cause de l'instabilité de l'IP.

Le meilleur moyen de protéger sa session d'un vol est de chiffrer la session entière grâce au SSL que nous allons voir un peu plus loin.

IV-B - Système de ticket

Un hacker qui vole une session signifie que l'utilisateur réel est lui aussi connecté sur le site, sinon il n'y aurait pas de session. Ce hacker volera une seule fois l'identifiant de session, puis l'utilisera.

En somme, il y a 2 utilisateurs sur le même compte.

L'idée est de donner un ticket à l'utilisateur qui demande une page. Pour obtenir la suivante, il faudra renvoyer ce ticket. Si celui-ci ne correspond pas, la session est annulée; si il correspond, un nouveau ticket est délivré.

Ainsi, voici le déroulement lors d'un vol de session

- Le visiteur demande une page
- On génère un ticket (exemple : toto123), on le mémorise coté serveur et on l'envoie coté client.
- L'utilisateur demande une nouvelle page
- On récupère le numéro du ticket coté client, on le compare avec celui en mémoire côté serveur
- Le numéro correspond, on envoie la page et on renouvelle le ticket (exemple : titi456)
- Le hacker demande ensuite la page (il connaît l'identifiant de session et possède un ticket qui lui est propre)
- Le ticket du hacker ne correspond pas avec celui de l'utilisateur, on coupe la session

En pratique, il suffit de créer un cookie que l'on renouvelle à chaque page, on le stocke aussi en session.

création du ticket

```
$ticket = uniqid(strval(mt_rand() (1, 999999)));
$_COOKIE['ticket'] = $ticket;
$_SESSION['ticket'] = $ticket;
```

Lors de l'appel d'une nouvelle page, on vérifie.

Vérification du token

```
if ($_COOKIE['ticket'] == $_SESSION['ticket'])
{
```


Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Vérification du token

```
    // On renouvelle
    $ticket = uniqid(strval(mt_rand() (1, 999999)));
    $_COOKIE['ticket'] = $ticket;
    $_SESSION['ticket'] = $ticket;
   }
   else
   {
    // On DECONNECT
   }
```

Cette technique présente tout de même une grosse faiblesse. Si le hacker vole la session alors que l'utilisateur vient de terminer sans se déconnecter, le hacker réussira.

En effet, le hacker vole l'identifiant de session et le ticket délivré au membre lorsque celui-ci appelle la dernière page, le hacker les utilise ensuite comme les siens. Pour le serveur, il n'y a aucun moyen de voir la permutation, vu qu'il ne voit qu'un seul utilisateur tout le long.

Il faut forcer du mieux possible l'utilisateur à se déconnecter manuellement après chaque session.

Vous vous dites peut-être que vous avez déjà vu l'idée quelque part.

*En effet, PHP définit la fonction **session_regenerate_id( )**.*

Cette fonction fait globalement la même chose que notre système de ticket, à la différence (énorme) que l'ancienne session n'est pas effacée et qu'elle ne s'applique pas à un ticket arbitraire mais à l'identifiant de session. Ainsi, chaque appel à session_regenerate_id ajoute une nouvelle session et augmente les chances de se faire voler une de ses sessions.

*Heureusement, depuis PHP 5.1.0, cette fonction prend le paramètre facultatif [bool delete_old_session]. Mais attention, **celui-ci est à false par défaut**.*

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

V - Chiffrement des données

Nous avons vu comment protéger le mot de passe lors de la connexion. Cependant, on peut vouloir protéger d'autres données que l'on souhaite garder en clair, comme l'email ou certaines informations personnelles. Pour cela, nous pouvons utiliser des algorithmes de chiffrement largement éprouvés.

V-A - Algorithmes existants

V-A-1 - Chiffrement symétrique

Aussi appelé "à clé privée" car il est fondé sur le principe simple qu'un code est indéchiffrable tant qu'on ne connaît pas la clé (normal ;-).

C'est le système de chiffrement le plus connu, il est vieux comme le monde et a servi à toutes les époques, pour tous les contextes : personnel, industriel, militaire (qui ne connait pas enigma ?), services secrets...

C'est aussi le plus simple à appréhender car fondamentalement, tout ce qu'on ne comprend pas est plus ou moins déchiffrable par un code, le meilleur exemple qui me passe par la tête étant l'ADN. Nous avons appris à lire pour en tirer un "code génétique", maintenant, nous écrivons grâce à ce code. L'inconvénient du chiffrement symétrique, c'est qu'il faut que le destinataire ET l'expéditeur connaissent le même code, sinon, comme pour l'ADN, il faudra chercher. Ils devront donc à un moment ou un autre communiquer ensemble de manière non chiffrée, au moins pour s'échanger le code de chiffrement.

Pour donner une image, ça correspondrait à envoyer par la poste la clé d'un coffre fort de banque.

Pour permettre l'échange de cette clé, on utilise le chiffrement asymétrique que nous allons voir maintenant.

Parmi les algorithmes de chiffrement symétrique, on trouve le DES, le tripleDES, le Blowfish, le RC2 à RC6, le IDEA...

V-A-2 - Chiffrement asymétrique

Aussi appelé "à clé publique", même si en fait, on manipule une paire de clés : une publique et une privée.

Il est asymétrique car ne fonctionne que dans un seul sens. Par exemple :

- A veut **recevoir** une information provenant de B.
- A génère une clé publique et une clé privée.
- A envoie la clé publique à B.
- B chiffre son information avec la clé publique.
- B envoie le message chiffré à A.
- A déchiffre en utilisant la clé privée.

La génération de ces clés est fondée sur la difficulté de factorisation des grands nombres.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

En gros, ça veut dire qu'on peut aisément multiplier 2 énormes nombres entre eux, mais qu'on ne peut pas retrouver si facilement ces 2 nombres en ayant que le résultat de cette multiplication.

En pratique, on choisi 2 grands nombres premiers. Il faut qu'ils soient premiers pour que leur multiplication ne donne qu'une seule factorisation possible.

Par exemple 9 et 6 sont très mauvais, car 9x6=54, or 54 est aussi égal à 27x2, 9x6, 3x18.

Les algorithmes de chiffrement asymétrique sont beaucoup plus gourmands que ceux symétriques, on les utilise donc pour chiffrer les informations essentielles, comme par exemple, la clé de chiffrement privée d'un chiffrement symétrique.

Les deux types d'algorithmes sont donc totalement complémentaires.

Parmi les algorithmes de chiffrement asymétrique, on trouve le RSA (le plus connu), le PGP, le DSA...

V-B - Cas pratique avec le RSA

Nous allons ici voir comment chiffrer une donnée provenant d'un formulaire utilisateur.

Tout d'abord, il convient de bien situer le but.

Je rappelle que cette partie concerne les données que l'on souhaite chiffrer coté client afin de ne pas les faire circuler sur le réseau en clair, mais qu'un hash n'est pas possible car nous avons besoin de cette donnée en clair coté serveur.

Chronologiquement, ça va se dérouler de la sorte.

Déroulement du chiffrement asymétrique d'une donnée distante

- Le serveur génère une paire de clés publique/privée.
- Il garde la clé privée (en session ou en base)
- Il envoie la clé publique avec le formulaire demandant la donnée à chiffrer.
- On chiffre côté client en utilisant la clé publique.
- On envoie le formulaire contenant uniquement la donnée chiffrée (on retire évidemment celle saisie par l'utilisateur en clair).
- On déchiffre côté serveur avec la clé privée pour récupérer notre donnée.

Nous avons donc besoin de générer la paire de clés en PHP, de chiffrer en Javascript et de déchiffrer en PHP.

Une clé est en fait constituée de 2 valeurs, un module et un exposant.

Coté PHP

Pour la partie PHP, nous utilisons le package Crypt_RSA (version 1.2.0b) de PEAR dont le lien est disponible en fin d'article.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Il utilise les extensions PHP de gestion de grands nombres. **BCMath** par défaut (la plus lente, mais la plus courante), **BigInt** ou **GMP** (classées ici de la plus lente à la plus rapide).

Elle est compatible PHP4 et PHP5.

J'y ai apporté quelques modifications. Ces modifications sont visibles dans les annexes.

Le but principal de ces modifications est de permettre la transmission de PHP vers JS et inversement en utilisant des nombres et non des hiéroglyphes.

Coté Javascript

À l'image du package Crypt_RSA de PEAR, j'ai utilisé une librairie de gestion de grands nombres auquel j'ai ajouter quelques fonctions nécessaires pour coller avec notre modèle en PHP.

Au final, on a 2 fonctions principales. Une pour charger notre clé publique créée par PHP, l'autre pour encrypter.

```
function RSAPublicKey(encryptionExponent, modulus, nbBits)
function encryptedString(key, s)
```

Utilisation

Procédons par ordre chronologique. Tout d'abord, il nous faut générer les clés. Si vous vous souvenez bien, ceci est la charge du serveur, donc de Crypt_RSA.

Code de génération des clés

```
set_include_path('./PEAR/');

session_start();

// Le package Crypt_RSA de PEAR
require_once 'Crypt/RSA.php';

// Les modifs apportées sur ce package
require_once './class/myKey.php';
require_once './class/myKeyPair.php';
require_once './class/myRSA.php';

$nbBits = 128;
$key_pair = new Crypt_RSA_KeyPair($nbBits, 'BCMath');
$public_key = $key_pair->getPublicKey();
$private_key = $key_pair->getPrivateKey();

// on met en session la clé privée
$_SESSION['private_module'] = $private_key->getModulus();
$_SESSION['private_exp'] = $private_key->getExponent();
```

Remarquez que nous mettons la clé privée en session.

Il nous faut maintenant passer la clé publique au Javascript.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Passage de la clé publique au js

```
<script language="JavaScript" src="./js/BigInt.js"></script>
<script language="JavaScript" src="./js/BigIntMath.js"></script>
<script language="JavaScript" src="./js/RSA.js"></script>
<script language="JavaScript">
var key;
var EncryptedText;

// La clé publique passées aux js
var public_exponent = "<?php echo $public_key->getIntExponent(); ?>";
var public_modulus =  "<?php echo $public_key->getIntModulus(); ?>";

var nb_bits = <?php echo $nbBits; ?>;

// Initialisation de BigInt
setMaxDigits(154);

// Création de la clé publique
public_key = new RSAPublicKey(public_exponent, public_modulus, nb_bits);
```

Nous définissons maintenant 2 fonctions, l'une pour récupérer la valeur saisie par l'utilisateur, la chiffrer et la replacer dans le champ (en modifiant légèrement les propriétés du champ au passage), l'autre pour annuler le chiffrement.

fonction de contrôle du formulaire

```
function encrypt()
{
    var code_str = document.register.codesecret.value;
    EncryptedText = encryptedString(public_key, code_str);
    document.register.codesecret.type='text';
    document.register.codesecret.size=EncryptedText.length;
    document.register.codesecret.value=EncryptedText;
    document.register.isCrypted.value=1;
}

function annule()
{
    document.register.codesecret.type='password';
    document.register.codesecret.size=15;
    document.register.codesecret.value='';
    document.register.isCrypted.value=0;
}
```

Enfin, il nous faut un formulaire utilisateur classique.

Formulaire utilisateur

```
<form id="register" method="post" action="" name="register">
    <fieldset>
     <legend>Saisissez votre code secret</legend>
  <div>
   <label>
    <strong>Code secret : </strong><br />
    <input type="password" name="codesecret" size="16" autocomplete="off"/><br />
   </label>
  </div>
  <br />
        <input type="button" value="Chiffrer votre code secret" onclick="encrypt()" class="bouton">
        <input type="button" value="Annuler le chiffrement" onclick="annule()" class="bouton">
        <input type="hidden" name="isCrypted" value="0">
    </fieldset>
    <p align="center"><input type="submit" name="register" value="Valider" class="bouton" /></p>
</form>
```


Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Rien d'exceptionnel, juste à noter un champ caché qui permettra au PHP de savoir si le champ est chiffré ou non.

Et pour finir, la récupération du formulaire, avec le déchiffrement du champ si besoin.

traitement du formulaire

```
if ( ! empty($_POST['codesecret']))
{
    $codesecret = $_POST['codesecret'];
    if ( (isset($_POST['isCrypted'])) && ($_POST['isCrypted'] == 1) )
    {
        $rsa_obj = new Crypt_RSA(null, 'BCMath');
        $private_key = new Crypt_RSA_Key($_SESSION['private_module'], $_SESSION['private_exp'],
 'private', 'BCMath');
        $codesecret = $rsa_obj->decrypt($codesecret, $private_key);
    }

    echo 'Votre code secret : '.$codesecret;
}
```

Testez ce script vous même.

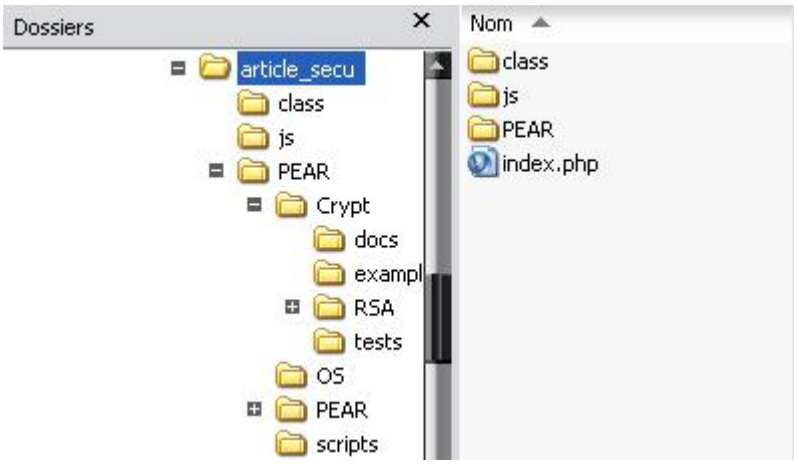
Pour reconstituer cet exemple, il vous faudra :

- [**Les sources de l'exemple**].

- [**Le package PEAR**]

- [**Le package Crypt_RSA**]

Renommez les répertoires de PEAR et Crypt_RSA de façon à obtenir cette arborescence.



Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

VI - Chiffrement de la session : HTTPS

VI-A - Présentation

Si vous avez lu attentivement jusqu'ici, c'est bon, vous avez tout pour comprendre le HTTPS.

Rien d'original en effet, du hash, du chiffrement symétrique et du chiffrement asymétrique.

Ainsi, une fois que le client et le serveur sont d'accord sur les algorithmes à utiliser, le serveur génère la clé publique et privée du chiffrement asymétrique, envoie la clé publique (en pratique, il s'agit plutôt de certificat) au client et garde la clé privée. Le client génère une clé pour le chiffrement symétrique, la chiffre avec la clé publique du serveur. Le tout est envoyé au serveur qui déchiffre avec sa clé privée.

Ce mécanisme permet ainsi de transmettre la clé privée d'un chiffrement symétrique de manière sécurisée.

Le HTTPS est en fait une utilisation du HTTP classique, sécurisé par SSL (Secure Socket Layer) inventé par Netscape.

Le SSL se place dans le modèle TCP au niveau de la couche session, sous la couche application.

Ceci est primordial car cela signifie que toute application peut utiliser SSL.

Voyons rapidement le mode de fonctionnement.

Lorsqu'on appelle un page en HTTPS, la première étape consiste pour le client et le serveur à se mettre d'accord sur les méthodes de chiffrement.

Pour se rendre compte de cet échange, on va utiliser Ethereal dont voici quelques captures.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

On voit dans la liste des informations envoyées par le client la liste des méthodes de chiffrement qu'il connaît :

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Le serveur va à son tour envoyer un "server Hello" avec plusieurs informations :

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Dans les informations qu'envoie le serveur, on trouve la méthode de chiffrement qu'il choisit, ainsi que son certificat (voir chapitre suivant).

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Enfin, la troisième phase, le client va générer la clé privée de chiffrement symétrique, la crypter grâce à la clé publique du serveur et l'envoyer au serveur.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

La transmission de données chiffrées grâce à cette clé peut maintenant commencer.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

VI-B - Déploiement

Pour utiliser le SSL sous windows, vous trouverez toutes les informations nécessaires en suivant [**ce lien**].

Pour linux, le mieux est de se renseigner auprès de la documentation de sa propre distribution, un package est bien souvent disponible.

VI-C - Certificat

Je viens de vous présenter le principe du HTTPS et ça a l'air incrackable.

Et bien tel que je viens de le présenter, ça ne vaut rien, car n'oublions pas que **depuis le début** de cet article, un homme est entre nous et le serveur.

Imaginez que cet homme vous envoie lui-même sa clé publique au bon moment et possède sa clé privée pour déchiffrer. Il sera capable de lire tout ce que vous écrivez comme s'il n'y avait aucun chiffrement.

L'important pour le hacker sera de faire croire au visiteur qu'il est bien connecté au bon site et non au hacker. Pour cela, il existe plusieurs techniques, on peut par exemple pirater le site ciblé et y mettre une redirection automatique vers le site du hacker qui sera un clone du site original. Pour résoudre ceci, il faut utiliser un certificat, qui consiste pour un serveur à enregistrer sa clé publique auprès d'un organisme externe qui ainsi certifie sa validité.

Le serveur n'envoie plus sa clé publique, mais le certificat entier que le client valide auprès de l'organisme correspondant.

En cliquant sur le petit cadenas en bas à droite de Firefox, vous pourrez ainsi voir ceci :

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

VII - Conclusion

Sécuriser ne veut rien dire.

D'une part, comme je l'ai mentionné, un procédé de chiffrement n'est fiable seulement que parce qu'il n'a pas encore été cracké. Il existe des organismes chargés de tenter de casser ces algorithmes et de grosses primes à ceux qui y arrivent.

D'autre part, mettre du HTTPS avec un certificat, rajouter sa couche de RSA, de GDS, de hash, *etc*. pour le site de l'association des joueurs de pétanques de sa ville, ça sert à rien.

Cet exemple montre que la sécurité, c'est relatif à la demande et aux informations contenues.

Il faut savoir que si un site perso, voire communautaire, se doit d'appliquer au moins une protection minimum des données de ses membres (donc hash+GDS), un site récupérant des informations comme un numéro de carte bleue ou de pièce d'identité doit **impérativement** utiliser le HTTPS.

Attention aussi à ne pas trop en mettre, l'illusion de la sécurité est encore pire que ne rien mettre. Vous vous devez de savoir quoi mettre et dans quel but.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

VIII - Annexes

VIII-A - Modifications du package Crypt_RSA de PEAR

Dans le but de modifier/ajouter des méthodes dans les classes, j'ai utilisé l'héritage (voir les classes du répertoire class de l'exemple d'utilisation du RSA). Les modifications réalisées sont les suivantes :

1) Ajout de deux fonctions dans la classe Key.

```
function getIntModulus()
{
    return $this->_math_obj->bin2int($this->_modulus);
}

function getIntExponent()
{
    return $this->_math_obj->bin2int($this->_exp);
}
```

Ces fonctions permettent de retourner le module et l'exposant de chaque clé sous forme de nombre, car à l'état initial après génération, on obtient un ensemble de signes ressemblant plutôt aux insultes de bandes dessinées, et ça passe plutôt mal sur une page web.

2) Modification de la "préparation" au déchiffrement réalisé par la méthode decrypt.

```
function decrypt($enc_data, $key = null)
{
 return $this->decryptBinary($this->_math_obj->int2bin($enc_data), $key);
}
```

Le but principal de ces modifications est de permettre la transmission de PHP vers JS et inversement en utilisant des nombres et non des "hiéroglyphes".

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

IX - Sources

Sources **Fichier Javascript de hash SHA1**

Sources **Fichier Javascript de récupération par Ajax du GDS**

Sources **Exemple d'implémentation du RSA pour le chiffrement de données coté client**

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

X - Liens

Interne

Exemple de craquage par force brute d'un mauvais mot de passe

Tutorial Ajax

Session Hijacking et IP spoofing

Installation Apache+OpenSSL+PHP sous windows

[Forum] Discussion sur la sécurité PHP

[Forum] Discussion sur la conception d'un espace membre PHP

[Forum] Un espace membre implémentant ce qui a été présenté ici

Externe

Implémentation SHA1 en Javascript

Exemple de deux formulaires pour un identification pour récupérer le GDS

Package RSA de Pear

Site très complet sur la cryptographie

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 25/01/2007 - Guillaume Affringue. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.