Sécurité et programmation côté serveur

TD R209

PHP et python peuvent être utilisés pour la programmation côté serveur. Dans les cas d'utilisation, on trouve par exemple le traitement de formulaires et enquêtes ou la gestion des mots de passe et de la base de données des utilisateurs. Dans la mesure où l'on va traiter sur le serveur des données entrées par l'utilisateur, il convient de prendre des précautions.

Stockage de mots de passe

Les mots de passe des utilisateurs sont stockés sur le serveur. La base de données des mots de passe peut être obtenue par des gens malveillants, il faut donc chiffrer les données. On utilise pour cela une "fonction à sens unique" nommée fonction de hashage, c'est-à-dire une fonction qui est facile à calculer mais difficile à inverser.

1. Considérons la fonction `md5`. Sur un ordinateur raisonnable, on peut "hasher" 10000 chaînes par seconde. Supposons que les mots de passe sont des chaînes de 8 lettres minuscules. Étant donné le hash, combien de temps cela prendra-t-il de trouver mon mot de passe ?
   a. Dans un premier temps, on calculera combien de mots de passe existent.
   b. Puis on calculera le temps total pour calculer les hashes de tous les mots de passe.
   c. Écrire un algorithme qui étant donné le hashé d'un mot de passe calcule le mot de passe.
   d. On supposera négligeable le temps de comparaison entre deux hashes. On obtient donc un temps assurant que le mot de passe est trouvé.
2. Supposons maintenant que les mots de passe sont constitués de majuscules, minuscules et de chiffres. Combien de temps pour déchiffrer le mot de passe à partir du hash ?
3. Même question en supposant que les mots de passe sont constitués de 10 caractères (majuscules, minuscules, chiffres).
4. Plutôt que trouver le mot de passe d'un utilisateur en particulier, je souhaite trouver un mot de passe quelconque parmi ceux des 1000 utilisateurs de mon site. Combien de temps cela prendra-t-il en moyenne de trouver un de ces mots de passe ?
5. On considère maintenant la fonction de hashage sha256. Celle-ci est plus lente. Sur la même machine, on peut "hasher" 100 chaînes par seconde. Mêmes questions que précédemment.
6. Une personne malveillante possède une base de données de hashes des 20% de mots de passe les plus fréquents. Il peut grace à cela rapidement trouver les mots de passe de 80% des utilisateurs du site. Pour parer à cela, on ajoute un "sel" propre au site à la chaîne avant de la hasher. Comment, sans connaître le sel a priori, la personne malveillante peut-elle tenter d'obtenir les mots de passe de certains utilisateurs ?

Données utilisateurs

Dans le cadre de formulaires, une application pourra lire des informations données par l'utilisateur et les afficher par la suite dans une page web.

Par exemple, si l'utilisateur entre dans le champ prenom le mot "Emmanuel", le code php suivant

```
<?php
echo("Bonjour " . $_REQUEST["prenom"]);
?>
```

écrira `Bonjour Emmanuel` dans le code html transmis au navigateur.

1. Que se passe-t-il si l'utilisateur met des balises dans son prénom ?
2. L'utilisateur peut-il mettre du javascript qui sera exécuté dans ce champ ?
3. Proposer une solution pour éviter que les balises html (et les scripts) ne soient interprétées. (penser aux cours de R109)

Protocole d'échange de vélo

Alice veut prêter son vélo à Bob mais ils ne peuvent pas se rencontrer (Alice dort de 19h à 8h et Bob dort de 7h à 20h). Ils conviennent d'un lieu où déposer le vélo mais s'ils le laissent sans antivol, celui-ci sera volé !

1. Pouvez-vous imaginer un protocole permettant de transmettre le vélo sans risque de vol en utilisant uniquement des antivols à clef ?
2. Même question mais cette fois, Alice et Bob ne connaissent pas de personne de confiance à qui confier la clef de l'antivol.

Protocole d'échange de clef : NSPK

L'échange de clefs est indispensable pour établir une transmission chiffrée. On utilise pour cela le protocole d'échange de vélo mais en ajoutant des contraintes pour éviter les attaques MitM.

— Alice (A), Bob (B) et Sophie (S, serveur de confiance) ont des couples clef publique/clef secrète notés K_P^A, K_S^A...
— S possède les clefs publiques de tout le monde, tout le monde possède la clef publique de S.
— On notera $\{M\}_{K_P^X}$ pour le message M chiffré avec la clef publique de X.
— N_A et N_B sont des nonces, c'est-à-dire des nombres aléatoires à usage unique générés par Alice et Bob respectivement.

Le protocole Needham-Schroeder Public Key est un protocole d'échange de clef célèbre à la fin duquel, en théorie, Alice et Bob peuvent communiquer de façon sure (chacun a la clef publique de l'autre et sait que personne ne peut écouter la conversation).

1. Vérifier que le protocole est cohérent (quand quelqu'un utilise une clef, il la possède bien)
2. Vérifier qu'à la fin, Alice et Bob ont bien la clef publique l'un de l'autre et peuvent donc échanger des messages chiffrés.
3. Bob est-il sur que c'est bien Alice qui lui écrit et inversement ?

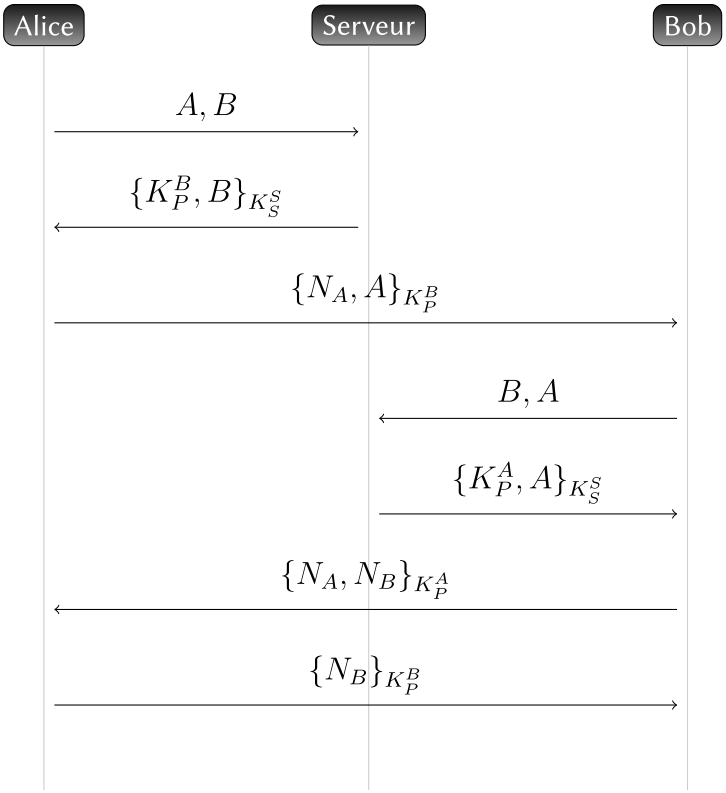

FIGURE 1 – NSPK

Syntaxe php

1. Écrire un script php qui écrit pour chaque carré entre 1 et 100 un paragraphe de la forme `<p>4 est un carré.</p>`.
2. Écrire un script php qui définit un annuaire téléphonique sous forme de tableau associatif et les affiche dans une table html.
3. Écrire une fonction php qui prend un nom comme entrée et renvoie le numéro de téléphone correspondant d'après l'annuaire défini à la question précédente.

Référence

```php
<?php
// Le code php doit être à l'intérieur d'une balise <?php ... ? >

// variables : $id
$x = 0;
$b = true;
$chaine = "Bonjour";

// Types et opérateurs :
// deux types numériques : les entiers et les flottants
$x = $x + 2 * 3 / 7;

// booléens et opérateurs not, or et and : ! || &&
$b = ! (true || false) && $b;

// L'opérateur de concaténation de chaines est le '.'
$chaine = $chaine . " à tous";

// La fonction d'affichage se nomme echo
echo("<p>" . chaine . "</p>");

/*********************************
 *       Conditionnelles         *
 *********************************/

if ($b) {
    $x = $x+1;
} else {
    $x = $x-1;
}

/*********************************
 *          Itérations           *
 *********************************/
// Itérations non bornées
while ($b) {
    $b = false;
}
```

```php
// Itérations bornées
for ($i = 0; $i < 17; $i = $i + 1) {
    echo($i);
}

/*****************************************
 *           Tableaux (arrays)           *
 *****************************************/
// initialisation
$t = array();
$u = array(7, 10, 33);

// cases numérotées à partir de 0
$t[0] = false;
$t[1] = true;
echo($t[0]);
echo(count($t)); //le nombre de cases de $t

// parcours d'un tableau avec for
for ($i=0; $i < count($t); $i = $i + 1) {
    echo("La case " . $i . " vaut " . $t[$i]);
}

/*****************************************
 *              Fonctions                *
 *****************************************/
// Définition de fonction
function nomDeFonction(a, b) {
    $x = $a + $b;
    return $x;
}

// Appel de fonction
$z = nomDeFonction(7, 12);

/*****************************************
 *       Tableaux associatifs            *
 *           (Hashtables)                *
 *****************************************/
// Déclaration
$d = array();
$dictionnaire = array(
    "alpha" => 1,
    "beta"  => 2,
    "omega" => 23,
);

// remplissage
$d["true"] = false;
```

```php
$d["alpha"] = "omega";

// accès
$chose = $dictionnaire["alpha"];
$truc  = $dictionnaire["beta"];

//Savoir si une case est définie : isset
if (isset($dictionnaire["gamma"])) {
    // $dictionnaire possède bien un case gamma
    echo("$dictionnaire["gamma"]");
} else {
    // $dictionnaire n'a pas de case gamma. Créons en une
    $dictionnaire["gamma"] = 33;
}

// parcours avec foreach
foreach ($dictionnaire as $cle => $valeur) {
    //$cle va parcourir les indices (clefs) du dictionnaire
    //$valeur va parallèlement parcourir les valeurs.
    echo("Dictionnaire " . $cle . " -> " . $valeur . " ou " . $dictionnaire[$cle]);
}
?>
```