M1 - Outils de l'Internet (Année 2010/2011)

TD n° 9 - Examen 2009-2010

Exercice 1. *Questions en vrac*

1. Expliquez ce qu'est le *Network Address Translation* ou NAT (que l'on pourrait traduire par « traduction d'adresse réseau » en français). Illustrez le fonctionnement par un exemple avec un schéma.
Pourquoi est-ce nécessaire avec des adresses en IPv4 (sur 32 bits), mais pas très utile avec des adresses en IPv6 (sur 128 bits) ?

2. Citez les points forts du langage HTML et ses inconvénients. Qu'est-ce que le XHTML apporte ?
Le langage *Portable Document Format* (PDF) développé par Adobe dans les années 90 avait été envisagé comme successeur du HTML pour décrire les pages web. À votre avis, pourquoi a-t-on conservé le HTML ?

3. Que sont les entrées DNS *Mail eXchanger* (MX) ? Expliquez quand et comment elles sont utilisées (par exemple si l'on veut envoyer un mail à `bob@example.com`).

4. Citez quelques méthodes pour lutter contre le spam (à différents niveaux : utilisateurs, serveurs mails, administrateurs réseaux, etc.)

5. Qu'est-ce que la *Common Gateway Interface* (CGI) ?

6. Quels sont les avantages et les inconvénients des applications web (par rapport aux applications classiques) ?
Citez trois exemples d'applications qui sont très adaptées à être mises sous forme d'applications web, et un exemple d'application qui ne pourrait pas être une application web.

7. Expliquez en quelques lignes le système des boîtes CSS.
Comment peut-on faire en HTML simple (sans CSS) pour placer un texte à 20 pixels du bord gauche de la page ?

8. Expliquez le fonctionnement des *Message Authentication Codes* (MAC) permettant à deux utilisateurs Alice et Bob de s'assurer qu'un message reçu par Bob provient bien d'Alice (authentification) et qu'il n'a pas été modifié par un utilisateur tiers pendant la transmission (intégrité).
Rappel : Alice et Bob ont chacun une clé publique et une clé privée, la clé publique de chacun étant connue de l'autre (et de tout le monde d'ailleurs).

9. Expliquez brièvement pourquoi le protocole WEP n'est pas fiable pour protéger un réseau Wi-Fi (donnez les idées principales des attaques vues en TD).

10. Expliquez comment différents sites peuvent se mettre daccord avec un tiers pour que ce dernier « piste » les utilisateurs à laide de cookies. Le but étant de déterminer pour chaque utilisateur lensemble des pages quil visite sur lensemble des sites concernés (par exemple pour mieux connaître les intérêts des utilisateurs et cibler la publicité).
Indication : Les sites placent un objet (par exemple une image) qui se trouve sur le serveur du tiers sur chacune de leurs pages pour que lutilisateur effectue une requête vers le tiers à chaque lecture dune page...

Exercice 2. *SMTP*

Supposons que l'on ait la conversation suivante entre un client et un serveur SMTP (les lignes précédées de `S:` proviennent du serveur, celles précédées de `C:` du client) :

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<danny@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: Elijah <elijah@example.org>
C: To: alice@example.com
C: Cc: Charlie <charlie@example.com>
C: Date: Tue, 15 Jan 2008 16:02:43 -0500
C: Subject: Test
C:
C: Bonjour !
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
```

1. Expliquez la différence de signification entre les deux lignes

```
C: RCPT TO:<alice@example.com>
C: To: alice@example.com
```

2. En supposant que tous les serveurs SMTP par lesquels le message est passé (et par lesquels il passera) sont correctement configurés et fiables, de quelle adresse provient le message, et à quelles adresses arrivera-t-il ?

3. Que doit-on reprocher au créateur du message ? Que croiront les personnes qui recevront le message ?

Exercice 3. *Redirection de POST vers GET*

1. Décrivez les différences entre les méthodes `POST` et `GET` lorsqu'il s'agit de transmettre des paramètres au serveur HTTP.

On considère maintenant un site web construit selon le modèle MVC.

2. Rappelez quelles sont les 3 parties du modèle MVC, et décrivez en une ligne la fonction de chacune.

Traditionellement on réserve la méthode `GET` aux requêtes qui ne modifient pas l'état du serveur (les paramètres servent simplement à préciser ce que l'on veut afficher). Les paramètres qui modifient les informations du serveur (changement dans la base de données) sont alors plutôt transmis par une requête de type `POST`.

3. Selon ce principe, justifier pourquoi il n'est jamais gênant d'envoyer plusieurs fois la même requête de type `GET` tandis qu'il peut être problématique de transmettre plusieurs fois une requête de type `POST`. Illustrez le problème (dans le cas d'une méthode `POST`) à l'aide d'une situation d'exemple.

4. Décrivez plusieurs situations dans lesquelles un utilisateur (n'ayant pas de connaissances sur le fonctionnement du protocole HTTP) pourrait émettre plusieurs fois la même requête `POST` sans s'en rendre compte.
indication : pensez aux différentes fonctionnalités (menus, boutons, etc.) proposées par votre navigateur préféré et imaginez des situations courantes dans lesquelles l'utilisateur pourrait les utiliser.

La plupart des navigateurs affichent un message d'alerte lorsqu'un utilisateur essaie de répéter une requête `POST` (pour s'assurer que c'est bien volontaire). Ce message est cependant en général trop obscur pour que l'utilisateur moyen le comprenne.
Afin de limiter les risques, de nombreux sites affichent eux-mêmes des mises en garde indiquant à l'utilisateur de ne pas effectuer certaines opérations pour assurer le bon déroulement du processus. Ces messages sont en général inquiétants et ne résolvent pas entièrement le problème...
La solution est alors d'utiliser une redirection de `POST` vers `GET` :

a. La requête `POST` est émise par le client, contenant des informations qui vont modifier l'état du serveur ;
b. La base de donnée du côté du serveur est modifiée ;
c. Le serveur redirige le navigateur vers une requête `GET` c'est-à-dire qu'il lui transmet une nouvelle URI (contenant des paramètres `GET`) ;
d. La requête `GET` est émise par le navigateur ;
e. Le serveur transmet la page qui correspond au résultat de l'opération qui a été effectuée par la requête `POST`.

5. Toutes les réponses HTTP contiennent un code numérique qui indique le type de la réponse. Citez les 5 types de réponses possibles (avec un bonus si vous donnez à chaque fois le code correspondant).

6. Expliquez quelles modifications il faut apporter du côté du serveur web pour effectuer une redirection de `POST` vers `GET`. Indiquez à quelle étape du fonctionnement du serveur elle peut être effectuée [1].

7. Expliquez ce qui se passe dans le cas d'une redirection de `POST` vers `GET` lorsque l'utilisateur recharge la page sur laquelle il arrive après avoir soumis le formulaire ou lorsqu'il utilise successivement les boutons « page précédente » et « page suivante ».

8. Expliquez pourquoi cette pratique correspond mieux à une conception MVC.

1. On rappelle que les étapes successives d'un serveur Apache sont, dans l'ordre : *Child Initialization*, *Post Read Request*, *Translate Name*, *Header Parser*, *Check Access*, *Check User ID*, *Check Auth*, *Type Checker*, *Prerun Fixups*, *Handlers*, *Logger* et *Child Exit*.