Spam

La bête aux multiples visages

- Français : *pourriel*, *polluriel*
- Anglais : *unsollicited bulk e-mail* (UBE), *junk e-mail*, *unsollicited commercial e-mail* (UCE)

Le phénomène

- Progression exponentielle depuis le début du mail
- Aujourd'hui on estime qu'environ 78% des messages reçus sont du spam (160 milliards par jour)
- Coûte environ 130 milliards de dollars par an (perte de productivité et coût technique)

Origine

- Le phénomène existait déjà sur les réseaux télégraphiques (depuis 1860)
- Le terme vient d'un sketch du *Monty Python Flying Circus*
- *SPAM = Spiced ham*

Sources d'adresses

- Les spammeurs récupèrent les adresses à l'aide de *crawlers* sur des sites, forums de discussion, virus (carnets d'adresses), etc.
- Véritable marché des adresses (entre spammeurs)
- Une grande partie des messages est envoyée à des adresses invalides
- Recoupement de bases de données

Contenu

- Publicité (pornographie, médicaments, crédits financiers, jeux d'argent en ligne, etc.)
- Propositions frauduleuses (héritages, transferts bancaires, emplois faciles, etc.)
- Lettres en chaîne
- Messages d'erreur suite à un message non envoyé
- Phishing

Fraude 419

- Scam nigérian
- Idée : attirer la cible par un gros gain, puis lui demander d'avancer des frais
- Gain estimé à plus de 100 millions par an
- 18e siècle : *Lettre de Jerusalem*, principe expliqué par Vidocq dans son livre *Les voleurs*.

Législation

- Europe : messages commerciaux non sollicités interdits par la *directive du 12 juillet 2002 sur la protection de la vie privée dans le secteur des communications électroniques*

- Article L.34-5 du code des postes et des communications électroniques : "*Est interdite la prospection directe au moyen d'un automate d'appel, d'un télécopieur ou d'un courrier électronique utilisant, sous quelque forme que ce soit, les coordonnées d'une personne physique qui n'a pas exprimé son consentement préalable à recevoir des prospections directes par ce moyen.*"

Législation (2)

- La récupération automatique d'adresses mail est également interdite (article 226-18-1 du code pénal)
- Les spammeurs exploitent fréquemment des vulnérabilités sur des machines tierces (indépendantes) → articles 323.1-3

Législation (3)

- Aux USA : CAN-SPAM Act, 16 décembre 2003
  (Controlling the Assault of Non-Solicited Pornography And Marketing)
- *Opt-out* : tout utilisateur doit pouvoir décider de ne plus recevoir les messages (en Europe : *opt-in*)
- Champs doivent être conformes au contenu
- Le message ne doit pas être envoyé par un *open relay*, l'en-tête ne doit pas être falsifiée

Falsification d'identité

- Création de fausses identités pour obtenir des comptes chez les FAI
- Utilisation de numéros de cartes de crédit volés
- Utilisation d'une société tierce pour transmettre les messages

Falsification d'identité (2)

- Falsification des adresses (*address spoofing*)
  - Problème de SMTP → utilisation de SMTP-AUTH (mais loin d'être suffisant)
  - Falsification du trajet du message (champ `Received` de l'en-tête)

Conséquences de l'*address spoofing*

- Messages d'erreur concernant des messages non-envoyés
- Utilisateurs "légitimes" peuvent être accusés de spam (plaintes, liste noire, résiliation du service du FAI)

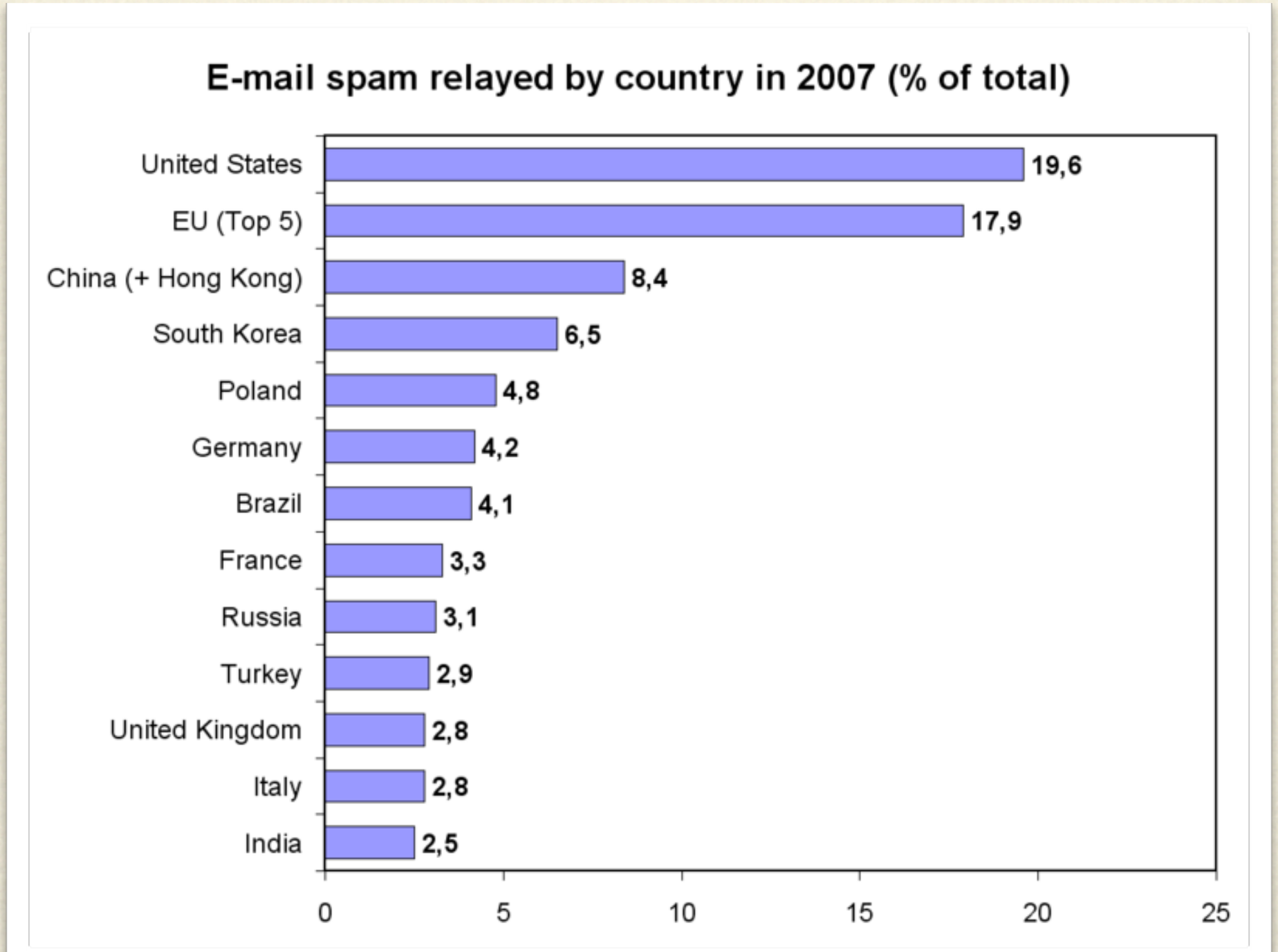
Exploitation de ressources extérieures

- *Open Mail Relays*
- Utilisation de *Proxys*
- Attaque d'ordinateurs à l'aide de vers ou virus → ordinateurs *zombis* (environ 80% des spams sont envoyés par des zombis)

Records

- D'après Steve Ballmer, Bill Gates reçoit 4 millions de mails par an (principalement du spam)
- Le domaine *acme.com* reçoit plus d'un million de spam par jour

Origine


E-mail spam relayed by country in 2007 (% of total)
United States
EU (Top 5)
China (+ Hong Kong)
South Korea
Poland
Germany
Brazil
France
Russia
Turkey
United Kingdom
Italy
India
19,6
17,9
8,4
6,5
4,8
4,2
4,1
3,3
3,1
2,9
2,8
2,8
2,5
0
5
10
15
20
25

Techniques anti-spam

Détection

- En fonction du contenu (attention aux faux positifs)
- Liste noire (spammeurs, open relays, zombis, etc.)
- Pièges (adresses non communiquées)
- Respect des normes (logiciels de spams sont souvent incomplets, pas assez de contrôle sur les ordinateurs distant, ...)

Utilisateurs

- Discrétion : ne partager son adresse qu'avec un groupe de confiance (attention aux *fwd*)
- Obscurcissement des adresses contre les bots
- Ne pas répondre au spam
- Utiliser des formulaires HTML de contact
- Désactiver le HTML dans les mails (failles de sécurité et contenu indésirable)

Utilisateurs (2)

- Utilisation d'adresses temporaires
- Mot de passe (dans le sujet ou l'adresse)
- Plainte (mais risque de se plaindre au spammeur)
- Exceptionnellement : répondre au spammeur (p.ex commentaires négatifs sur un site)

Administrateurs

- *Bloquage* (en fonction de l'origine) et *filtrage* (en fonction du contenu)
- Authentification et confiance des serveurs
- Défi/Réponse
- Comparaison des messages par *checksum* (pour repérer les messages envoyés en masse)

Pour les administrateurs (2)

- Filtrages géographiques
- Liste grise (rejet temporaire des messages de serveurs inconnus)
- Modification des enregistrements MX
- Jeux sur les normes

Pour les administrateurs (3)

- Certificats payants de non-spam
- Pots de colle (réponses très lentes aux serveurs inconnus)
- Test anti-virus des pièces jointes

Filtrage du contenu

- Expressions régulières (mots clés interdits)
- Filtres statistiques (bayésiens)

Hameçonnage (*phishing*)

Description

- Obtention d'informations personnelles (identifiants, mots de passe, etc.) en se faisant passer pour une entité de confiance
- Généralement par e-mail ou messenger

Histoire

- Premier cas documenté en 1987
- Apparition du terme *phishing* en 1996

Le cas AOL

- 1995 : mesures d'AOL pour interdire l'utilisation de numéros de CB générés automatiquement
- Provoque des tentatives d'obtenir des numéros réels
- Un escroc se fait passer pour un administrateur d'AOL et demande à un utilisateur d'entrer ses informations...
- 1997 : AOL prend des mesures importantes contre le phishing. Fin de l'incident.

Attaque sur les milieux financiers

- 2001 : nouvelles tentatives dirigées contre des organismes bancaires
- Echecs… mais la pratique mûrit
- 2004 : pratique criminelle répandue
- En 2006, la moitié des tentatives provient du *Russian Business Network*.

Masquage de liens

- La principale technique est de pointer vers un site frauduleux par un lien qui semble légitime :
  - `http://www.google.faux.com/`
  - `<a href=faux>vrai</a>`
  - `http://www.vrai.com@www.faux.com/`
  - utilisation de noms de domaines internationalisés

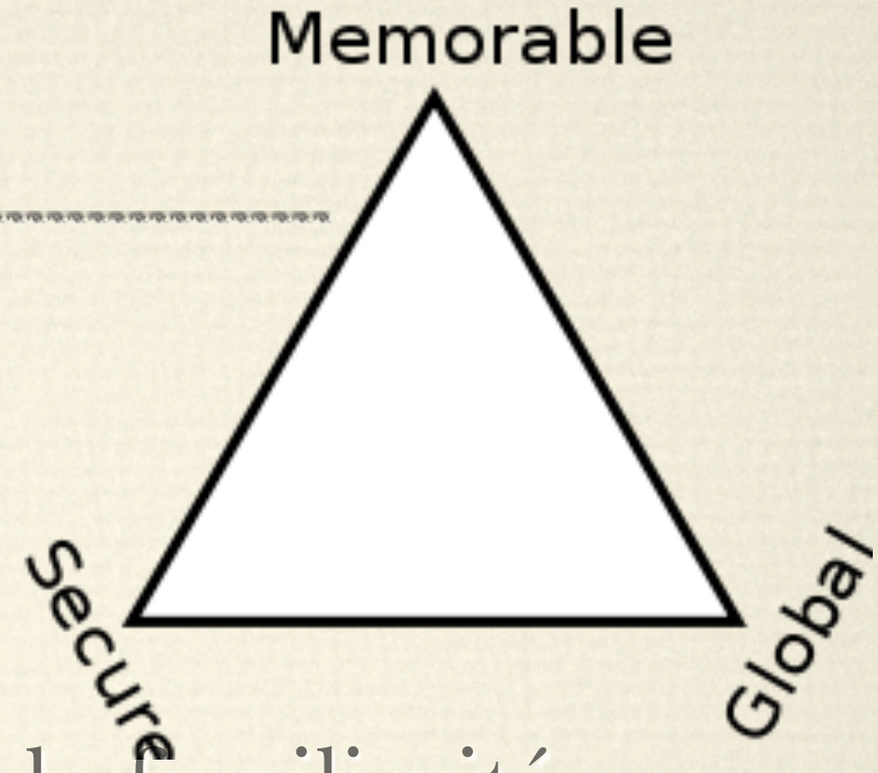
Autres techniques

- Utilisation d'images pour éviter la reconnaissance de texte
- Falsification d'un site à l'aide de *Javascript*
- Exploitation de failles dans un site légitime (*cross-site scripting*)

Solutions

- Sensibiliser les utilisateurs au problème (apprendre à identifier les tentatives)
- Rendre l'authentification plus claire (*pet names*)
- Identifier les sites frauduleux
- Personnaliser les communications légitimes
- Eliminer les mails d'hameçonnage

Pet Names



- Triangle de Zooko
- Compromis entre la sécurité et la familiarité
- Idée : utiliser des noms choisis par l'utilisateur pour faire le lien
- *keys*, *nicknames* et *pet names*

4. Le web dynamique

M1 Outils de l'Internet
lundi 12 octobre 2009

victor.poupet@lif.univ-mrs.fr

Web statique

Le web statique

- Le client veut une ressource (`http://www.blop.com/index.html`)
- Il transmet une requête au serveur (`GET`)
- Le serveur envoie la ressource (`200 OK`)
- Le client voit la ressource (Firefox)
- Le client veut une nouvelle ressource (`http://www.blop.com/autre.html`)

Le web statique

- HTML
- XML
- XHTML
- CSS
- C’est bien mais on s’ennuie vite

Les idées qui marchent bien

- Adressage (identification par URI)
- Formats structurés et adaptés : HTML, CSS, SVG, PNG, etc.
- Protocole simple et efficace (HTTP)

Un peu d'interactivité

- Les formulaires (`POST` ou `GET`)
- Les sessions (`POST`, `GET` ou cookies)

Le web à double sens

- La vision initiale de Tim Berners-Lee permettait de lire et d'écrire sur le web
- La capacité d'écriture disparaît très vite
- Jim Whitehead décide en 1996 de retrouver cet aspect → WebDAV

WebDAV

- Web based distributed authoring and versionning
- Groupe de travail de l'IETF
- Extension de HTTP

WebDAV

- Nouvelles méthodes :
  - `PROPFIND` Obtenir les propriétés (XML)
  - `PROPPATCH` Modifier les propriétés
  - `MKCOL` Créér une collection
  - `COPY`
  - `MOVE`
  - `LOCK` Mettre des verrous
  - `UNLOCK` Supprimer des verrous

Alternatives

- FTP
- CVS et SVN
- SMB et Samba
- Wiki
- ...

Web dynamique

Le web dynamique

- Idée : document vu par le client dépend de paramètres extérieurs
- 2 possibilités :
  - côté client
  - côté serveur

Côté client

Côté client

- Le contenu dynamique est généré sur l'ordinateur du client au cours de la présentation
- Aucun travail de la part du serveur
- Le navigateur exécute le script

Côté client

- Inconvénients :
  - le navigateur peut ne pas connaître le langage
  - l'information n'est accessible qu'au client
  - les moteurs de recherche ne peuvent pas exécuter le script
  - risques de sécurité pour le client

JavaScript

- Développé par Netscape (*Mocha*, *LiveScript* puis *JavaScript*)
- Presque aucun rapport avec Java
- Conçu pour être facile d'utilisation pour les non-programmeurs

JavaScript

- dynamique
- faiblement typé
- orienté prototype
- fonctions de première classe
- syntaxe héritée du C (boucles, conditions, etc.)

Programmation dynamique

- Typage dynamique
- Orienté objet
- Tableaux associatifs : `blip.nom="blop"`
- Interprété

Prototypes

- Programmation objet à base de prototypes (par opposition aux classes)
- Construction d'objets avec les fonctions
  `new f`
- Les fonctions servent de méthodes
  `obj.f`

Caractéristiques

- evaluation à l'exécution (*runtime*)
- fonctions d’arité variable (objet `arguments`)
- expressions régulières

Syntaxe

"JavaScript borrows most of its syntax from Java, but also inherits from Awk and Perl, with some indirect influence from Self in its object prototype system." - Brendan Eich

- Les variables n'ont pas de type associé
- Elles ont la portée de la fonction où elles sont créées si l'on emploie le mot-clé var
- Sinon elles sont globales

Syntaxe

- Les commentaires sont indiqués par // ou /*, */
- JavaScript est sensible à la casse
- Les ; en fin de ligne ne sont pas obligatoires mais ils sont conseillés (ajout automatique : source de confusion)

Variables : exemple

```
x = 0; // variable globale
var y = 'blop'; // variable globale

function f(){
  var z = 'blip'; // variable locale
  nombre = 42; // variable globale
  return x;
}
```

Types fondamentaux

- Nombres : 42, 3.14, 1.664e2, 0x33, etc.
  (tous les nombres sont considérés comme *double*)
- Chaînes de caractères

```
var chaine1="bonjour";
var chaine2='au revoir';
```

  - les chaînes ne peuvent pas être modifiées

Types fondamentaux

- Tableaux (ont un attribut `length`)

```
myArray = [0,1,2,3,4,5];
myArray = new Array(0,1,2,3,4,5);
myArray = new Array(365);

cats = [{"color":"brown", "size":"large"},
        {"color":"black", "size":"small"}];
cats[0]["size"];

dogs = {"rover":{"color":"brown", "size":"large"},
        "spot":{"color":"black", "size":"small"}};
dogs["spot"]["size"];
```

Objets

- Fonctionnent comme des dictionnaires :

```
var obj = {nom: "un objet", utile: false, reponse: 42};
var un_nom = obj.nom;
var une_reponse = obj["reponse"];

var personne = {
  nom: {prenom: "John", nom: "Smith"},
  age: 42,
  hobbies: {"foot", "ping pong"}};
```

Boucles

- ```
  if (test) {
     ... ;
  } else if (test) {
     ... ;
  } else {
     ... ;
  }
  ```

- ```
  while (test) {
     ... ;
  }
  ```

Boucles

```
switch (expr) {
   case VAL:
      ... ;
     break;
   case VAL:
      ... ;
     break;
   default:
      ...;
     break;
}
```

Fonctions

- nom, paramètres, bloc d'instructions, retour :

```
function f(a, b, c){
   ... ;
   return res;
}
```

- Eventuellement pas de paramètre, voire pas de nom :

```
var val = function(a, b){
   ... ;
   return res;
}
```

Fonctions

- Le nombre d'arguments à l'appel peut ne pas correspondre au nombre d'arguments nommés dans la définition :
  - Si un argument nommé n'est pas donné il a la valeur `undefined`
  - Les arguments non nommés sont accessibles dans `arguments` (p.ex. `arguments[3]`).
  - arguments a un attribut `length` (mais ce n'est pas un tableau)

Fonctions : exemple

```
function pgcd(a, b) {
  while (a != b) {
    if (a > b) {
      a -= b;
    } else {
      b -= a;
    }
  }
  return a;
}
```

Création d'objets

- Déclaration

```
var monObjet = new Object();
```

- Initialisation

```
var objetA = {};
var objetB = {"nom": "un_objet",
"valeur": 42};
```

Création d'objets

- Constructeurs

```
function MonObjet(att1, att2) {
  this.attribut1 = att1;
  this.attribut2 = att2;
}

objet = new MonObjet(12, "chien");
```

L’héritage par les prototypes

```
function voiture() {
  this.nb_roues = 4;
  this.marque = "indéfinie";
}

function smart() {
  this.date = 1990;
  this.marque = "Renault";
}

clio.prototype = new voiture();
```

En pratique

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
  <head><title>Un joli titre</title></head>
  <body>
    <script type="text/javascript">
      document.write('Youpi tralala');
    </script>
    <noscript>
      Your browser does not support JavaScript.
    </noscript>
  </body>
</html>
```

Compatibilité

- Le JavaScript est exécuté par le client (navigateur)
- L'interface DOM de manipulation des pages n'est pas normalisée dans JavaScript
- Il existe une norme du W3C...
- ... mais elle n'est pas respectée par les navigateurs

Compatibilité

- On peut essayer de se limiter à du JS "standard"
- On peut tester si certaines fonctionnalités sont disponibles et agir en conséquence
- Il est possible que le script ne soit pas interprété du tout : en tenir compte

Côté serveur

Côté serveur

- Le client émet une requête HTTP
- Le serveur identifie le script ou application à appeler
- Le script est exécuté avec des paramètres provenant de la requête HTTP
- La sortie du script est transmise au client

Avantages

- Le serveur a une liberté totale de manipulation
- Le travail du script est transparent aux yeux du client
- Pas de problème de compatibilité
- Pas de risque pour le client
- Les utilisateurs n'ont pas accès au source (éventuellement un inconvénient)

Inconvénients

- Risques du côté serveur
- Plus de travail côté serveur

Nécessité d'exclure les bots pour alléger (`robots.txt`) :

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /images/
Disallow: /tmp/
Disallow: /private/
```

Common Gateway Interface

- Protocole commun d'interface entre une application externe et un serveur web
- Définit la façon dont les arguments sont passés à l'application externe (arguments et variables d'environnement)
- Définit la façon dont les informations sont renvoyées au serveur (en-têtes)

Common Gateway Interface

- Certaines adresses sont associées à une application externe
- Lorsqu'une URI correspondant à ces adresses est demandée, le serveur appelle l'application externe en lui transmettant les données du client
- La sortie de l'application est transmise au serveur, ainsi que les en-têtes nécessaires

PHP

- PHP: Hypertext Preprocessor
- Langage de scripts conçu pour générer des pages web dynamiques
- Implémentation principale par le PHP Group sert de norme (pas de spécification formelle)
- Version actuelle : 5.2.5 (nov 2007)

PHP

- PHP1 et 2 : créé en 1994 par Rasmus Lerdorf (alors nommé *Personal Home Page Tools*)
- PHP3 : réécriture du parseur en 1997 (rebaptisé selon l'acronyme récursif actuel)
- PHP4 (2000) : encore corrigé
- PHP5 (2004) : nombreuses améliorations
- PHP6 : en cours de développement, *unstable*

Utilisation

- Insertion dans du code HTML
- Entrée : PHP
- Sortie : HTML
- PHP est gratuit et compatible avec tous les serveurs web et systèmes d'exploitation
- Script exécuté dans un environnement contrôlé (*sandbox*)

Délimiteurs

- Seul le code dans le délimiteur PHP est interprété
- Délimiteurs : `<?php`, `?>` ou `<script language="php">`, `</script>` ou encore `<?`, `<?=` et `?>`
- Exemple :

```
<?php
   echo "Hello World!\n";
?>
```

Inclusion dans du HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//
EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <title>
   <?php echo htmlspecialchars($page_title);?>
  </title>
  </head>
  <body>
  <p>Hello</p>
  </body>
</html>
```

Syntaxe générale

- Les noms de variables sont sensibles à la casse et préfixées d'un `$`
- Les noms de classes et fonctions ne sont pas sensibles à la casse
- Commentaires : `/*`, `*/`, `//` et `#`
- Les espaces et retour à la ligne ne sont pas significatifs
- Les instructions sont terminées par `;`

Types de données

- Entiers
- Double précision
- Booléens
- Chaînes de caractères
- Objets (depuis la version 3)
- Tableaux (hétérogènes)
- Null
- Ressources

Les variables

- Le type dépend de la valeur et non de la variable
- On peut insérer des variables dans les variables :

```
$animal = "chien";
$phrase = "cet animal est un $animal.";
```

Chaînes de caractères

- Délimitées par simple ou double quotes
- Chaînes heredoc :

```
$un_texte = <<<RAVEN
   Once upon a midnight dreary,
   while I pondered weak and weary
RAVEN
```

- Concaténation avec .

Include

- Permet d'insérer un fichier dans un script PHP
  `<?php include("script.php") ?>`
- Le fichier est inclus par un préprocesseur en l'excluant du bloc PHP : le fichier peut être un bloc de HTML, ou un script PHP encadré dans des délimiteurs.

Conditions

- Syntaxe semblable au C :

```
<?php
   if( $age == 18 )
   echo 'Vous avez 18 ans';
   else
   echo 'Vous n'avez pas 18 ans';
?>
```

- Utiliser des accolades en cas d'instructions multiples

Opérateur ternaire

- Version compacte de `if ... else` :

```
if ($test == 1)
   echo "blop";
else
   echo "blip";
```

est équivalent à

```
$test == 1 ? echo "blop":echo "blip";
```

Boucles

- Comme en C...

```
<?php
   for ($i = 0; $i<10; $i++)
   echo "$i <br />\n"
?>
```

- Parcours d'un tableau avec `foreach`

```
$tableau = ("un"=>"one", "deux"=>"two",
"trois"=>"three");
foreach($tableau as $valeur)
   echo "$valeur <br />\n";
```

POST et GET

- Les valeurs transmises dans un formulaire par POST ou GET sont accessibles dans les tableaux `$_POST` et `$_GET`

```
<form action="page.php" method="post">
<select name="champ1">
...
<input name="champ2" type="text" />
...
```

POST et GET

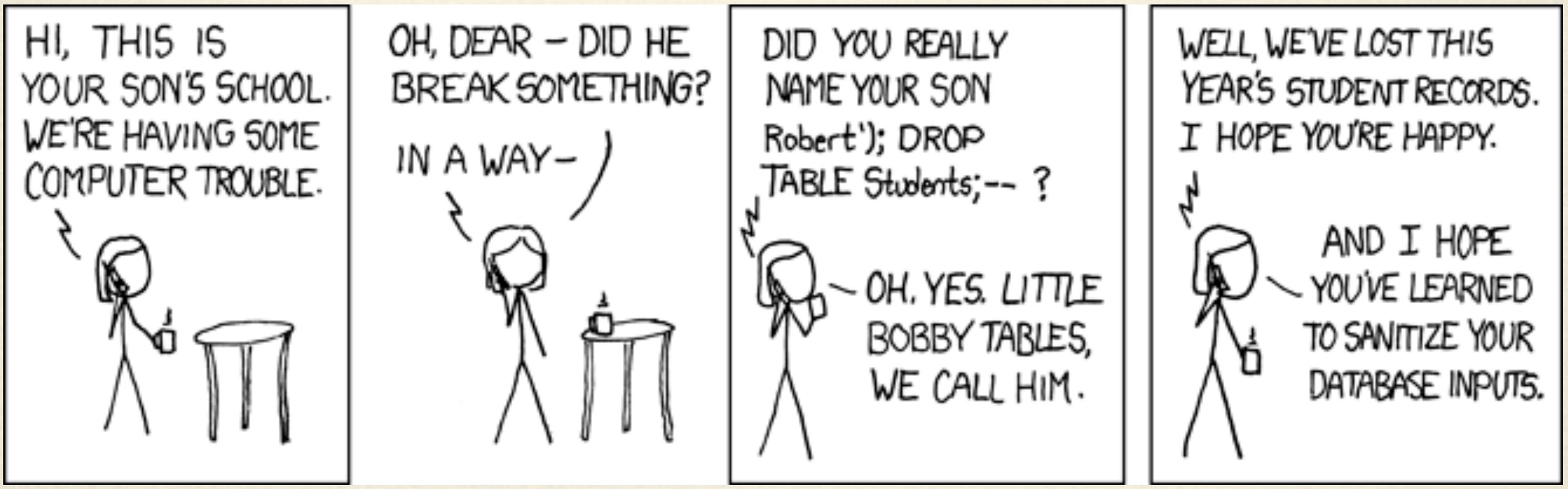
```
<?php
$var1 = $_POST["champ1"];
$var2 = $_POST["champ2"];
?>
```

- Même chose avec GET mais les paramètres sont passés dans l'adresse : `?champ1=blip&champ2=blop`

```
<?php
$var1 = $_GET["champ1"];
$var2 = $_GET["champ2"];
?>
```

POST et GET

- Attention ! Vérifier que les chaînes transmises par l'utilisateur sont valides...



http://xkcd.com/327/

Entités HTML

- Pour éviter d'interpréter une chaîne de caractères, on peut utiliser la fonction `htmlentities` :

```
<?php
   $entree=
   "<script type='text/javascript'>
   window.location:'http://www.evil.com/'
   </script>";
   $propre=htmlentities($entree);
   echo $propre;
?>
```

Sessions

- Il est possible de créér des sessions pour chaque utilisateur (`session_start()`)
- On peut alors stocker des variables dans le tableau `$_SESSION`
- L'utilisateur est identifié à l'aide de cookies ou par un paramètre envoyé par POST ou GET mais les variables sont enregistrées côté serveur
- Risque de vol de session

Cookies

- Le langage PHP contient de quoi gérer les cookies
  - `<?php setcookie(nom, valeur, expire, chemin, domaine, securite); ?>`
  - On récupère la valeur du cookie dans `$_COOKIE["nom"]`

Fichiers

- Il est possible de manipuler des fichiers texte avec PHP (fichier txt = BDD du pauvre)

```
if (! ($f=fopen("exemple.txt","r")))
exit("Unable to open file!");
```

- Différents modes disponibles : `r`, `r+`, `w`, `w+`, `a` , `a+`, `x`, `x+`.

Fichiers

- Lecture :

```
fread($f, $taille);
```

- Ecriture :

```
fwrite($f, $texte);
```

Interface MySQL

- Une vraie BDD c'est mieux qu'une pile de fichiers texte
- Des fonctions pour se connecter à une BDD SQL :

```
$connexion = mysql_connect(hote,
login, mdp);
mysql_close($connexion);
```

Des requêtes

- On choisit la BDD avec :
  `mysql_select_db(nom);`
- On fait des requêtes :
  `mysql_query(requete_SQL, connexion);`
- On peut utiliser les fonctions `CREATE`, `ALTER` et `DROP`