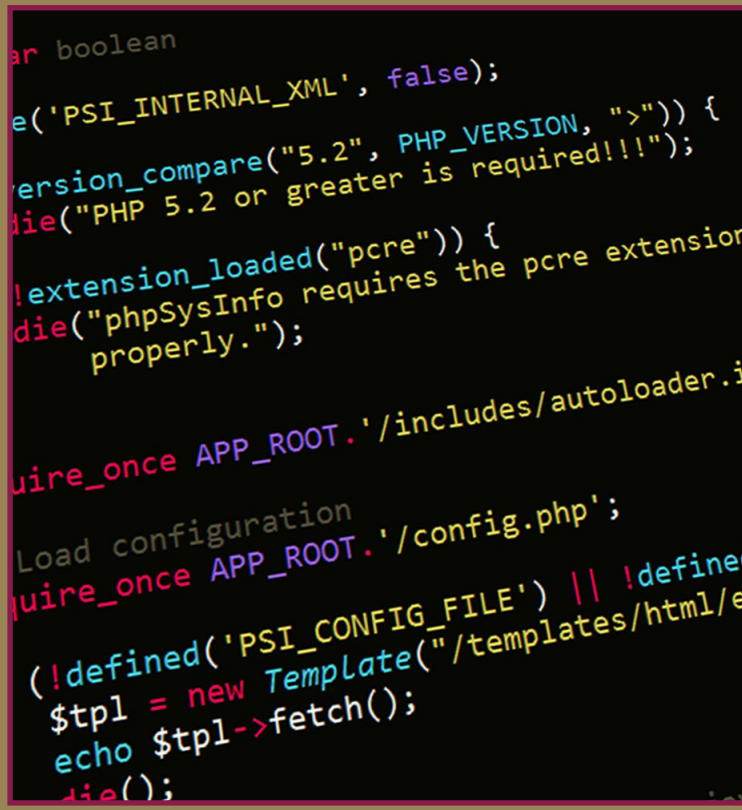
13

Protéger son site web des attaques Sql Injection



CENTRE DE RECHERCHE
SUR L'INFORMATION
SCIENTIFIQUE ET TECHNIQUE
cerist

PROTÉGER SON SITE WEB DES ATTAQUES SQL INJECTION

Les attaques de type injection sont considérées comme l'une des attaques les plus critiques. Dans les documents OWASP TOP10 des années 2007, 2010 et 2013, ce type d'attaque occupe toujours les toutes premières positions.

Une injection SQL fait en général référence à une attaque qui cible les applications web qui interagissent dynamiquement avec une source de donnée, dans le cas usuel, une base de données.Cette attaque est réalisée en injectant du code SQL dans une entrée utilisateur dans le but de former une nouvelle requête SQL et qui est non prévue par le programmeur. Cela permet à un attaquant de récupérer des données sensibles à partir de la base de données (récupérer des mots de passe par exemple).

Une telle attaque cible une faille dans une application web. Ce type de faille peut exister si les entrées utilisateur (formulaires, URL ou tout entête HTTP) ne sont pas filtrées correctement et sont utilisées comme paramètres pour récupérer des informations à partir de la base de données.

Exemple de faille : page d'authentification

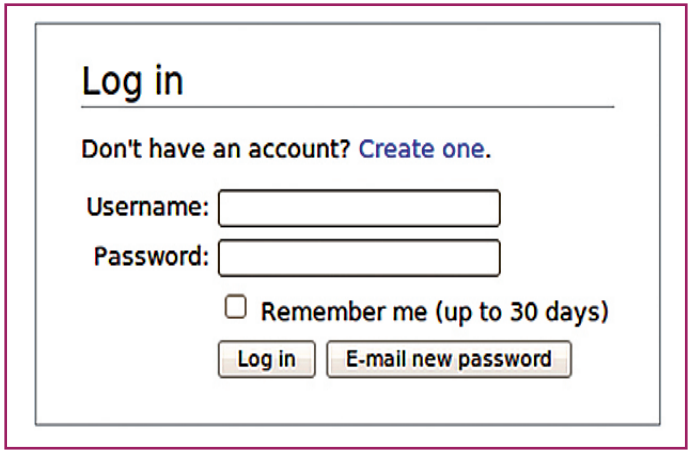
```php
<?php

...

mysql_connect(...);
mysql_select_db(...);

...

//récupération des paramètres
$utilisateur=$_POST['user'] ;
$password   =$_POST['password'] ;

//construction de la requête SQL
$requete='SELECT * FROM user WHERE user='. $utilisateur . 'AND password='. $password ;

//exécution de la requête
$result = mysql_query($requete);
...

?>
```

Nous illustrons un exemple de faille par une page d'authentification. Cette page contient deux champs : le champ utilisateur et le champ mot de passe. Ces deux données sont passées à un code PHP qui va vérifier leur légitimité par rapport à une base de données :

texte :
< ?php
…
mysql_connect(…);
mysql_sqlectdb(..) ;
…
//récupération des paramètres

```
$utilisateur=$_POST[‘user’] ;
$password=$_POST[‘password’] ;
//construction de la requête SQL
$requete=’SELECT * FROM user WHERE user=’. $utilisateur . ‘AND password=’. $password ;
//exécution de la requête
mysql_query($requete) ;
…
?>
```

Ici, la requête SQL est construite à partir des données envoyées par le formulaire. Aucune modification ou traitement sur ces données n’est effectué.



Exemple d’exploitation : contournement du mécanisme d’authentification

Le code précédent contient une vulnérabilité de type SQL injection. Les paramètres du formulaire sont envoyés tels quels. L’attaquant peut injecter du code SQL dans l’un des champs du formulaire. Il peut par exemple insérer:

Utilisateur : admin
Mot de passe : ‘ or 1=1 --
La requête devient après construction :
SELECT * FROM user WHERE user=’admin’ AND password=’’ or 1=1 --

La condition sur le mot de passe est toujours évaluée à vraie. L’attaquant peut donc se connecter avec l’identifiant admin sans vraiment connaître le mot de passe.

Classification des injections SQL :

Outre l'exemple précédent, plusieurs variantes des attaques
par injection SQL existent, elles peuvent être classées comme suit :

Tautologies : le principe ici est d'injecter un code pour que les instructions conditionnelles soient toujours évaluées à vraie (exemple du contournement du mécanisme d'authentification).

Requête avec union : le but est d'exploiter un paramètre de la requête pour extraire des données d'une table (autre que celle normalement utilisée dans la requête).

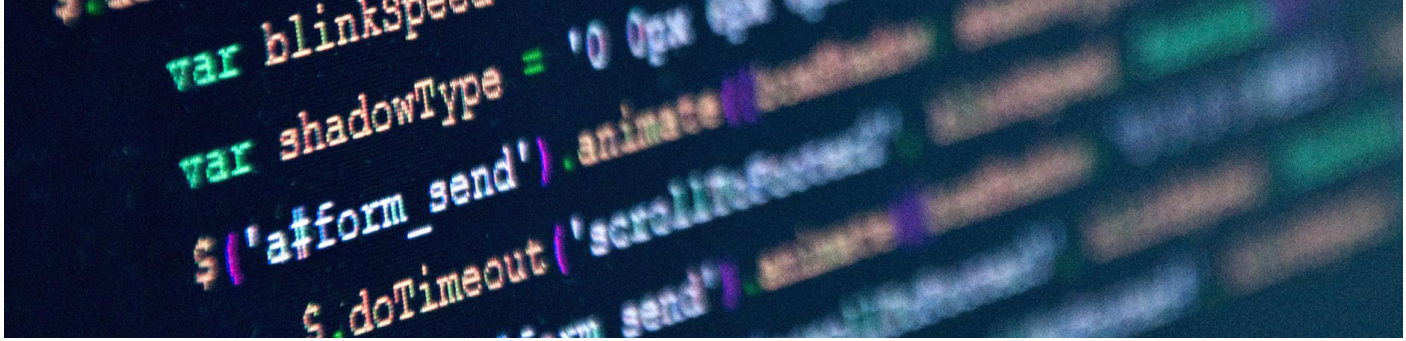
Requête Piggy-Backed : l'attaquant essaie avec ce type d'attaque d'injecter une nouvelle requête sans modifier la requête originale.

Requête incorrecte : elle consiste à causer intentionnellement une erreur dans la requête afin de récolter des informations importantes sur le système, l'application web ou la base de données.

Procédures stockées : le principe est d'exécuter les procédures stockées fournies par le SGBD. Dans les SGBD modernes, ces procédures permettent même l'interaction avec le système d'exploitation.

Inférence : consiste à modifier la requête pour simuler un mécanisme de réponse vrai ou faux. Comme le site ne retourne aucun message, l'attaquant doit observer le changement des pages du site web afin d'en déduire la réponse.

Encodage alternatif : est utilisé conjointement avec les autres techniques. Son but principal étant d'éviter la détection par les mécanismes de défense existants.



Techniques de protection :

Plusieurs techniques existent pour mitiger le risque des attaques par injection SQL.
Leur utilisation est grandement recommandée voir même obligatoire. Il est très important de prendre l'habitude de les utiliser notamment pour les développeurs.

Les procédures stockées : en utilisant cette technique, les données entrées par l'utilisateur sont transmises comme paramètres, sans risque d'injection.

Les expressions régulières : utiliser ces dernières permet de s'assurer que les données entrées par l'utilisateur sont bien de la forme souhaitée.

Principe moindre privilège : utiliser le principe du moindre privilège afin de limiter les droits des utilisateurs de l'application web ainsi que ceux de la base de données et ainsi limiter les risques ou les dégâts en cas d'intrusion.

Les requêtes paramétrées : en utilisant une requête paramétrée, c'est le SGBD qui se charge d'échapper les caractères selon le type des paramètres.

Les fonctions d'échappement : utiliser les fonctions d'échappement (exemple des fonctions mysql_real_escape_string et addslashes dans PHP) afin de filtrer les caractères spéciaux.