Injection Sql

En quoi consiste une injection sql ?

Dans le cadre des processus logiciels standard, une requête SQL est essentiellement une demande envoyée à une bas
e de données (un référentiel d'informations informatisé)
pour un certain type d'activité ou de fonction telle qu'une requête de données ou l'exécution d'un code SQL.
C'est par exemple le cas lorsque les informations de connexion sont envoyées via un formulaire Web pour permettre
à un utilisateur d'accéder à un site.
Généralement, ce type de formulaire Web est conçu pour accepter seulement des types de données très spécifiques, t
els que le nom et/ou le mot de passe.
Lorsque ces informations sont ajoutées, elles sont vérifiées par rapport à une base de données et, si elles corresponde
nt, l'utilisateur est autorisé à entrer.
Dans le cas contraire, il est privé d'accès.
Des problèmes potentiels peuvent survenir, car la plupart des formulaires Web ne disposent d'aucun moyen pour arr
êter la saisie d'informations
supplémentaires sur les formulaires. Les pirates peuvent exploiter cette faille et utiliser les boîtes de saisie du formul
aire pour envoyer
leurs propres demandes vers la base de données. Cela pourrait potentiellement leur permettre d'effectuer plusieurs ty
pes d'activités malveillantes,
allant du vol de données sensibles à la manipulation d'informations de la base de données à leurs propres fins.

Comment détourner une injection sql ?

Les canaux d'entrée utilisateur étant le principal vecteur d'attaques par injection SQL, la plupart des méthodes défen
sives impliquent
le contrôle et la validation de la saisie de l'utilisateur pour les modèles d'attaque.
Voici plusieurs mesures pouvant garantir la sécurité des entrées de l'utilisateur.

1- JAMAIS CONFIANCE EN LA SAISIE DE L'UTILISATEUR
La première règle empirique concernant la saisie de l'utilisateur est « ne faites pas confiance et ne vérifiez pas »,
ce qui signifie que toutes les formes de saisie de l'utilisateur doivent être considérées comme malveillantes, sauf pre
uve du contraire.
Cela ne concerne pas seulement les zones de saisie simples, telles que les zones de texte et les zones de texte, mais é
galement tout le reste:
les entrées masquées, les paramètres de chaîne de requête, les cookies et les téléchargements de fichiers.

Le fait que l'interface utilisateur du navigateur ne permette pas à l'utilisateur de manipuler une entrée ne signifie pas

qu'elle ne peut pas être falsifiée. Des outils simples tels que Burp Suite activé Les utilisateurs doivent capturer les
requêtes HTTP
et modifier n'importe quoi, y compris les valeurs de formulaire masquées, avant de les soumettre au serveur.
Et si vous pensez être malin en codant vos données en Base64, celles-ci peuvent facilement être décodées, modifiées
et recodées par des utilisateurs malveillants.

2- Valider les chaînes d'entrée côté serveur
La validation consiste à s'assurer que le bon type d'entrée est fourni par les utilisateurs et à neutraliser toute comma

nde malveillante
potentielle pouvant être intégrée dans la chaîne d'entrée. Par exemple, en PHP, vous pouvez utiliser mysql \ _real \ _ escape \ _ string ( )
pour échapper des caractères susceptibles de modifier la nature de la commande SQL.

Une version modifiée du code de connexion mentionné précédemment serait la suivante:

$ con = mysqli_ connect ("hôte local", "utilisateur", "mot de passe", " db ");

$ username = mysqli_real_escape_ string ($ con, $ _POST ['nomutilisateur']);

$ password = mysqli_real_escape_ string ($ con, $ _POST ['password']);

$ sql_command = "select * from users où username = ' ". $ nom d'utilisateur; $ sql_command . = "'AND password ='". $ mot de passe. """;

Cette simple modification protégerait votre code contre l'attaque présentée en ajoutant un caractère d'échappement (\) devant les guillemets simples ajoutés
intentionnellement par l'utilisateur malveillant.

Une note sur la validation : Si vous avez ajouté des fonctions de validation côté client, bravo. Mais ne vous fiez pas à cela comme mesure de défense contre les attaques par injection SQL. Bien que les fonctions côté client puissent re ndre plus
difficile l'envoi d'entrées malveillantes sur votre serveur, il est facile de les contourner avec quelques modifications du
navigateur et des outils tels que celui qui vient d'être mentionné. Par conséquent, vous devez le compléter par une va lidation côté serveur.

3- Utiliser les paramètres de commandes

Une meilleure alternative à l'échappement serait d'utiliser des paramètres de commande. Les paramètres de comman de sont définis en ajoutant des noms d'espaces réservés
dans les commandes SQL, qui seront ultérieurement remplacées par une entrée utilisateur. #ASP.NET dispose d'un ensemble d'API très intuitif et facile à utiliser à cette fin.

Signaler une annonce

SqlCommand cmd = new SqlCommand ("SELECT * FROM utilisateurs WHERE nomutilisateur = @ nomutilisate ur ET mot de passe = @ mot de passe" , con);

SqlParameter username = new SqlParameter ( ); nom_utilisateur.NomParamètre = "@nom_utilisateur"; username.v alue = txtUsername.Text ; cmd.Parameters.Add (nom d'utilisateur);

SqlParameter password = new SqlParameter ( ); password.ParameterName = "@password"; password.value = txtPa ssword.Text ; cmd.Parameters.Add (mot de passe);

Vous commencez par créer un SqlCommand objet et en utilisant le @ paramètre_name the paradigme dans la chaî ne de commande où l'entrée utilisateur doit être insérée.

Vous créez ensuite des instances de SqlParameter objets, dans lesquels vous insérez l'entrée utilisateur, au lieu de la concaténer directement avec la chaîne de commande.

Enfin, vous ajoutez le SqlParameter objecter à la SqlCommand La collection Parameters de l'objet, qui remplac era les paramètres avec l'entrée fournie. ADO.net s'occupe du reste.

En PHP, l'équivalent est préparé, ce qui est un peu plus compliqué que son équivalent ASP.net. Vous pouvez l'expl orer ici (https://www.php.net/manual/en/mysqli-stmt.bind-param.php)

4-EXPLICITE EXPLICITEMENT VOTRE ENTRÉE
Cette astuce concerne les langages tels que PHP, qui sont faiblement typés, ce qui signifie que vous ne définissez gé néralement pas les types de données pour les variables, et que le langage se charge automatiquement de convertir dif férents types de données entre eux.

Les conversions explicites peuvent servir de raccourci pour échapper des entrées lorsque des types non-chaînes sont impliqués. Donc, si vous attendez de l'utilisateur qu'il entre un int pour le age paramètre, vous pouvez assurer la sécurité de l'entrée avec le code suivant en PHP:

```
$ age = ( int ) $ _POST ['age'];
1
$ age = ( int ) $ _POST ['age'];
```

Notez que cet extrait de code valide uniquement le type de l'entrée, pas sa plage. Vous devrez donc exécuter un autr e code pour vous assurer que l'utilisateur n'entre pas d'âge négatif –
ou peu réaliste, tel que 1300.

En outre, une autre bonne pratique consiste à éviter d'utiliser des guillemets simples dans les commandes SQL dans lesquelles une entrée non chaîne est impliquée.
Donc au lieu d'utiliser le code suivant…

```
$ sql_command = "select * from users où age = ". $ age;
1
$ sql_command = "select * from users où age = ". $ age;
```

… Il serait un peu plus sûr d'utiliser le suivant:

```
$ sql_command = "select * from users où age = ' ". $ âge. """;
1
$ sql_command = "select * from users où age = ' ". $ âge. """;
```

Conclusion:
La bonne nouvelle est que la protection contre l'injection consiste «simplement» à filtrer correctement votre entrée e t à déterminer si l'on peut faire confiance à une entrée.
Mais la mauvaise nouvelle est que toutes les entrées doivent être filtrées correctement à moins de pouvoir faire con fiance à tout le monde (mais le dicton «ne jamais dire jamais»
vient ici à l'esprit).

Par exemple, dans un système à 1 000 entrées, filtrer 999 d'entre elles avec succès n'est pas suffisant, car il ne reste qu'un champ qui
peut servir de solution de rechange au traitement d' Achille . Et vous pourriez penser que l'insertion du résultat d'un e requête SQL
dans une autre requête est une bonne idée, car la base de données est fiable, mais si le périmètre ne l'est pas,
l'entrée provient indirectement de personnes atteintes de malintent . C'est appelé Deuxième injection SQL au cas où cela vous intéresserait.

Comme le filtrage est assez difficile à faire correctement (comme la crypto), ce que je conseille généralement est de

vous fier aux fonctions
de filtrage de votre infrastructure: elles ont fait leurs preuves et fonctionnent minutieusement. Si vous n'utilisez pas de framework,
vous devez réfléchir sérieusement à la question de savoir si ne pas leur utilisation est logique dans le contexte de s écurité de votre serveur. 99% du temps,
ce n'est pas le cas.

En d'autres termes, l'injection SQL existe depuis des décennies et continuera probablement de figurer en tête des tableaux de vulnérabilité pour les années à venir. Cela prend quelques étapes simples – mais bien calculées – po ur
vous protéger et protéger vos utilisateurs, et cela devrait faire partie de vos principales priorités lors de l'audit de votre code source pour détecter les failles de sécurité.

La clé pour éviter d'être victime de la prochaine grande violation de données d'injection SQL consiste tout d'abord à contrôler et à valider les entrées de l'utilisateur et, deuxièmement, à vous préparer au «quand» et non au « si».

Comment savoir si mon appli ou site est vulnérable a une injection sql ?

Étant donné que la base de données SQL (langage de requête structuré) est prise en charge par de nombreuses plates -formes Web (PHP, WordPress, Joomla, Java, etc.),
elle pourrait potentiellement cibler un grand nombre de sites Web. Donc, vous voyez, il est essentiel de vous assurer que votre site Web d'entreprise en ligne
n'est pas vulnérable à SQLi, et ce qui suit vous aidera à en trouver.

importante: L'exécution d'une injection SQL génère une bande passante réseau élevée et envoie beaucoup de donnée s.
Assurez-vous donc que vous êtes le propriétaire du site Web que vous testez.

1 - suIP.biz

Détection des failles d'injection SQL en ligne par suIP.biz prend en charge les bases de données MySQL, Oracle, Po stgreSQL, Microsoft SQL, IBM DB2, Firebird, Sybase, etc.

2 - Test d'injection SQL en ligne

n autre outil en ligne par Cible du pirate basé sur SQLMap pour trouver lier & erreur vulnérabilité basée sur la requê te HTTP GET.

3 - Acunetix

Acunetix est un scanner de vulnérabilité des applications Web prêt pour l'entreprise, approuvé par plus de 4000 mar ques dans le monde. Pas seulement l'analyse SQLi,
mais l'outil est capable de trouver plus de 6000 vulnérabilités.

Quelle est la prochaine?
Les outils ci-dessus vont tester et vous indiquer si votre site Web présente une vulnérabilité d'injection SQL. Si vous vous demandez comment protégez votre site contre l'injection SQL, alors ce qui suit vous donnera une idée.

L'application Web mal codée est souvent responsable de l'injection SQL, vous devez donc corriger le code vulnérabl

e. Cependant, une autre chose que vous pouvez faire est de mettre en œuvre le WAF (pare-feu d'application Web) de vant l'application.

Il y a deux possible comment intégrer WAF à votre application.

Intégrer WAF dans le serveur Web - vous pouvez utiliser WAF comme ModSecurity avec Nginx, Apache ou WebK night avec IIS.
Cela serait possible lorsque vous hébergez votre site Web seul, comme dans Cloud / VPS ou dédié.
Cependant, si vous êtes sur un hébergement partagé, vous ne pouvez pas l'installer là-bas.
Utiliser le WAF basé sur le cloud - probablement, le moyen le plus simple d'ajouter une protection de site consiste à implémenter le .
La bonne chose est que cela fonctionnera pour n'importe quel site Web et que vous pouvez le démarrer moins de mi nutes 10.

Les pratiques a prendre pour éviter une injection Sql :

Vous pouvez mettre en place différentes mesures pour empêcher une attaque par injection SQL de votre système de base de données.
Par conséquent, il est nécessaire de s'attarder sur l'ensemble des composants impliqués, tels que le serveur, sur chacune des applications mais aussi sur les systèmes de management de base de données.

Etape 1 : superviser la saisie automatique des applications

Testez et filtrez les méthodes et paramètres utilisés par les applications connectées lors de vos saisies dans la base de données.
Les données remises devraient toujours être en accord avec le type de données attendu. Si un paramètre numérique e st requis,
vous pouvez le vérifier avec un script PHP à l'aide de la fonction is_numeric(). En ce qui concerne les filtres, les caractères spéciaux correspondants sont ignorés. Un autre point important est de s'assurer que les applications n'émettent pas de messages d'erreurs externes qui indiquent des informations sur le système utilisé ou les structures de la base de données.

Entre-temps, d'autres pratiques se sont largement répandues, comme ce que l'on nomme les Prepared Statements qui

peuvent être utilisées avec de nombreux systèmes de management de banques de données. Ces déclarations prédéfin ies
étaient à l'origine utilisées pour exécuter des requêtes fréquentes, mais du fait de leur structure, elles permettent également de réduire les risques d'injection SQL. En effet, les instructions paramétrées transfèrent la commande SQ L
réelle des paramètres séparément de la base de données. Seul le système de management de la base de données lui-m ême $
relie les deux et masque ainsi automatiquement les caractères spéciaux importants.

Etape 2 : s'assurer de la protection complète du serveur

La sécurité du serveur sur lequel a cours votre système de gestion de base de données joue naturellement un rôle primordial dans la prévention contre les injections SQL. La première étape est donc de renforc er votre système d'exploitation selon le schéma établi :

Installer ou activer seulement les applications et services qui sont pertinents pour faire fonctionner la base de donnée s.
Supprimer tous les comptes utilisateurs qui ne sont pas nécessaires.
S'assurer que toutes les mises à jour des systèmes et programmes pertinents sont installés.

Plus votre projet Web exige des mesures de sécurité importantes, et plus vous devrez envisager
l'utilisation en amont de systèmes de détections d'intrusion (IDS) ou de systèmes de prévention d'intrusions (IPS).
Cela fonctionne avec différents systèmes de reconnaissance pour identifier en amont les attaques sur le serveur,
émettre des avertissements et dans le cas d'IPS de faire déclencher automatiquement les contre-mesures corresponda
ntes.
Un processus ALG (Application Layer Gateway) peut également être une mesure de protection judicieuse, en survei
llant
le trafic de données entre les applications et le navigateur Web directement au niveau de l'application.

Etape 3 : renforcer les bases de données et utiliser des codes sûrs

Tout comme votre système d'exploitation, les bases de données doivent être débarrassées de tout élément inutile
et être mises à jour régulièrement. A cette fin, éliminez toutes les procédures enregistrées dont vous n'avez pas
besoin et désactivez tous les services et comptes utilisateurs inutiles. Installez un compte spécifique pour votre base
de données,
qui sera prévu simplement pour y accéder depuis le Web, et qui sera pourvu de droits d'accès très restreints. Enregist
rez par ailleurs
dans votre base de données toutes les données sensibles de manière chiffrée, comme par exemple vos mots de passe.

Pour les Prepared Statements, il est urgemment recommandé de ne pas utiliser le module PHP mysql mais de choisir
à la place mysqli ou PDO.
De cette manière, vous pourrez par la même occasion vous protéger avec des codes sûrs. La fonction mysqli_real_es
cape_string() du script PHP
permet par exemple d'éviter d'émettre des caractères spéciaux aux bases de données SQL sous leur forme originale.

Ou se mettre a jour ?

Voici un site se tenir a l'actualité sur les injections sql :

- Silicon.fr (https://www.silicon.fr/tag/injection-sql)