

LES INJECTIONS SQL

Julien Condomines

SIO 1

TABLE DES MATIERES

Qu'est-ce que l'injection SQL ?	2
Découverte de l'injection SQL	2
Exemples d'attaques par injection SQL	3
Code Commentés.....	4
Modules d'insertion SQL	4
Relation Visual / MySQL	5
Forme de Relation Visual / MySQL	7
Comment se protéger d'une attaque par injection SQL ? ..	8
La validation de données	8
Utilisé un compte utilisateur	8
Les requêtes paramétrées	9
Conclusion.....	10

QU'EST-CE QUE L'INJECTION SQL ?

L'injection SQL est une forme de cyberattaque qui consiste à utiliser un morceau de code SQL pour entrer dans une base de données, donc accéder aux données normalement cachées dans la bdd, voire avoir la possibilité de modifier la base de données en elle-même, que ce soit sa structure ou l'identité des utilisateurs, si on envoie la bonne injection.

DECOUVERTE DE L'INJECTION SQL

Les injection SQL ont été découvertes en 1998, avec notamment Jeff Foristal, l'un des premiers à avoir documenter le sujet. Pendant qu'il écrivait un guide pour s'introduire sur un serveur Windows NT, il a découvert la faille et à proclamer pouvoir changer la façon dont SQL marchait.

EXEMPLES D'ATTAQUES PAR INJECTION SQL

Les attaque par injection SQL sont très courantes, dus à la simplicité de leurs utilisations, elles sont encore considérées aujourd'hui comme l'une des attaques informatiques les plus dangereuses.

On peut prendre pour exemple l'attaque de novembre 2005 par un adolescent, qui à l'aide d'une injection SQL a récupéré les informations clients d'un site de magazine du groupe The Tech Target.

En décembre 2009, un attaque à l'encontre de RockYou a été réalisé à l'aide d'injections SQL. L'attaquant réussira à s'introduire pour récupérer les nom d'utilisateurs et les mots de passe non cryptés de plus de 32 millions d'utilisateurs.

En octobre 2015, plus de 156 000 clients de la société de télécommunication TalkTalk's se sont vus dérobés leurs informations personnelles à l'aide d'une attaque sur la page web du site.

CODE COMMENTES

Modules d'insertion SQL

```
' signal au projet qu'il sera connecté à une bdd '  
Imports MySql.Data.MySqlClient  
0 références  
Module ModulePourEchangeDeVariable  
    ' connection a la bdd '  
    Public MaConnection As New MySqlConnection("Server=localhost;Database=sakila;Uid=root;pwd=")  
    ' v qui permet de lire la première commande sql '  
    Public MonReader As MySqlDataReader  
    ' v qui permet de lire la seconde commande sql '  
    Public MonReader2 As MySqlDataReader  
End Module
```

Injections SQL

Relation Visual / MySQL

```
' signal au projet qu'il sera connecté à une bdd '  
Imports MySql.Data.MySqlClient  
2 références  
Public Class Acteur  
    ' var qui n'affiche que le numéro d'acteur '  
    Public NumActeur As Int32  
  
    0 références  
    Private Sub Acteur_Load(sender As Object, e As EventArgs) Handles MyBase.Load  
        MaConnection.Open() ' permet de se connecter à la bdd  
  
        lblStatus.Text = MaConnection.State.ToString ' permet d'afficher le status de la connction à la bdd '  
  
        ' commande sql qui permet de montrer la table actor '  
        Dim MaCommandeSql As New MySqlCommand("Select actor_id,first_name, last_name from actor", MaConnection)  
  
        MonReader = MaCommandeSql.ExecuteReader() ' attribut à Mon Reader le droit de lire la commande sql '  
  
        dgvActeur.Columns.Add("id", "ID") ' crée des colonnes id nom et prénom dans un tableau '  
        dgvActeur.Columns.Add("nom", "Nom")  
        dgvActeur.Columns.Add("Prenom", "Prenom")  
  
        ' tant que la commande est active, affiche le résultat dans les colonnes créer précédemment et dans la combobox '  
        While (MonReader.Read())  
            cboNomEtPrenomActeur.Items.Add(MonReader.GetString(0) + " " + MonReader.GetString(1) + " " + MonReader.GetString(2))  
            dgvActeur.Rows.Add(MonReader.GetString(0), MonReader.GetString(1), MonReader.GetString(2))  
        End While  
  
        MaConnection.Close() ' marque la fin de la commande et déconnecte de la bdd '  
    End Sub
```

Injectons SQL

0 références

```
Private Sub ButtcmdClickLeave_Click(sender As Object, e As EventArgs) Handles ButtcmdClickLeave.Click
    Application.Exit() ' permet de fermer l'application '
End Sub
```

0 références

```
Private Sub dgvActeur_click(sender As Object, e As DataGridViewCellEventArgs) Handles dgvActeur.CellClick

    ' affiche l'id de l'acteur selectionner '
    lblNumActdgv.Text = dgvActeur.Rows(dgvActeur.CurrentRow.Index).Cells(0).Value
    ' le transforme en entier pour qu'il soit lisible '
    NumActeur = Convert.ToInt16(lblNumActdgv.Text)

End Sub
```

0 références

```
Private Sub cboNomEtPrenomActeur_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cboNomEtPrenomActeur.SelectedIndexChanged

    ' v prend la valeur de la combobox et la convertie en string '
    Dim ValeurRetournee As String = cboNomEtPrenomActeur.SelectedItem.ToString
    ' affiche dans un label l'acteur selectionner dans la combobox '
    lblActeur1st.Text = cboNomEtPrenomActeur.SelectedItem

    ' représente le nombre de films jouer par l'acteur selectionner '
    lblNbCar.Text = InStr(cboNomEtPrenomActeur.SelectedItem, " ")
    ' permet de couper les données reçu pour n'afficher que ce que l'on souhaite '
    lblNumActeur1st.Text = ValeurRetournee.Substring(0, Convert.ToInt16(lblNbCar.Text) - 1)

End Sub
```

```
Private Sub cmdResearch_Click(sender As Object, e As EventArgs) Handles cmdResearch.Click
    dgvFilms.Columns.Clear() 'permet de reinitialiser les colonnes d'affichage '
    MaConnection.Open() ' se connecte à la bdd '

    Dim Films As String = txtFilms.Text ' v pour commandes sql '
    lblStatus2.Text = MaConnection.State.ToString ' affiche l'état de la conncion à la bdd '

    ' commande sql pour afficher l'id, le titre, la description en fonction de ce que l'on rentre dans la textbox '
    Dim MaCommandeSql3 As New MySqlCommand("Select film_id, title, description from film where title like '%" & Films & "%' or description like '%" & Films & "%'",
    lblCommandeSQL.Text = MaCommandeSql3.CommandText ' affiche la commande sql dans un label '

    MonReader2 = MaCommandeSql3.ExecuteReader() ' attribut à Mon Reader le droit de lire la commande sql '

    dgvFilms.Columns.Add("id", "ID") ' créer des colonnes id, title, description dans le tableau '
    dgvFilms.Columns.Add("title", "Title")
    dgvFilms.Columns.Add("description", "Description")

    While (MonReader2.Read()) 'tant que la commande est active, affiche le résultat de la commande sql dans les collones du tableau'

        dgvFilms.Rows.Add(MonReader2.GetString(0), MonReader2.GetString(1), MonReader2.GetString(2))

    End While

    MaConnection.Close() ' ferme la connection à la bdd '
End Sub
```

Injections SQL

Forme de Relation Visual / MySQL

Acteurs

Choisissez l'acteur souhaité :

Recherche de films

Status

Commande SQL

Status

Numéro + Nom acteur :

Acteur

Numéro acteur :

Numéro acteur

Nombre de films :

Numéro Car

Exit

COMMENT SE PROTEGER D'UNE ATTAQUE PAR INJECTION SQL ?

La validation de données

C'est l'une des méthodes les plus basiques, elle consiste à valider les données renseignés par l'utilisateur.

A l'avance, il va falloir marquer sans le code le type de données accepter, chaîne de caractère ou nombre. Si l'utilisateur ne rentre pas le type demander, alors refuser la demande.

On peut contrôler la bonne saisi des données de plusieurs manière, a l'aide de booléens, de date mais aussi en contrôlant la longueur ou encore en vérifiant les domaines des adresses mails saisies.

Utilisé un compte utilisateur

Lorsque l'on code, on créer souvent la base de données à l'aide du compte par défaut (ROOT), ce qui peut être une source de faille car il a accès à toute la base et toute les tables en lecture et en écriture.

On peut donc pour limiter cela créer un utilisateur qui n'a les droits qu'en lecture sur une partie de la table, afin d'éviter toutes les dérives possible due au compte ROOT.

Les requêtes paramétrées

Ce sont des requêtes SQL qui ont reçu des marqueurs de place, afin de s'assurer que la requête SQL garde la forme souhaité lors de son exécution.

Exemple d'implantation

```
<source style="border: 1px dashed #2f6fab;background-color: #f9f9f9;font-size: 127%;padding: 1em;" lang="sql">
reqParam = SELECT * FROM user WHERE lastname = ? AND
firstname = ? </source>
```

Dans cet exemple, on crée une requête paramétrée qui retournera les informations des utilisateurs ayant un nom et un prénom précis. Pour intégrer les valeurs à cette requête, on peut utiliser les marqueurs de place comme suit :

```
<source style="border: 1px dashed #2f6fab;background-color: #f9f9f9;font-size: 127%;padding: 1em;" lang="sql">
reqParam.setParam( 1, "Alice" ); reqParam.setParam( 2, "Case" ); </source>
```

CONCLUSION

Pour conclure, on peut dire que l'injection SQL, bien que ce soit une attaque connue des développeurs, peut encore servir de nos jours aux pirates informatiques sur des sites peu sécurisés, comme notamment les sites Free ou Skyrock, gérés par des utilisateurs peu habitués au développement qui ne connaissent pas ce genre d'attaque.