SQLインジェクション Ver 1.1

2025 年 10 月 13 日

1　準備作業

1.1　init.sql を読み込む

work フォルダの init.sql を読み込む。work フォルダで ターミナルを開き、MariaDB に root でログインして、init.sql を読み込む。
その際、init.sql の文字コードが Shift_JIS (CP932) であることを確認するように。

```
1 MariaDB [(none)]> \. init.sql
```

すると、次のようなデータベース・テーブルが作成される。

データベース	userdb
ユーザー名	userdbuser
パスワード	userdbuser

テーブル名 members

id	INT	PRIMARY KEY, AUTO_INCREMENT
name	VARCHAR(20)	NOT NULL
password	CHAR(6)	NOT NULL

初期データ

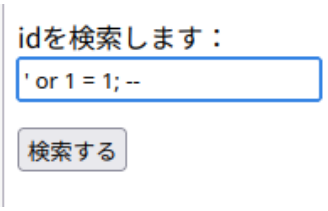
id	name	password
1	田中太郎	111111
2	佐々木次郎	222222
3	木村花子	333333
4	山本百合子	444444

2　未対策のプログラムコードを実行する

2.1　実行の手順

“work¥before¥” フォルダをドキュメントルートとしてサーバーを起動する。index.php と search.php が動作する。

index.php にアクセスすると、以下のように表示されるので、 ' or 1 = 1; -- と入力する。



すると、以下のように出力される。

id「' or 1 = 1; -- 」であるレコード

ID	名前	パスワード
1	田中太郎	111111
2	佐々木次郎	222222
3	木村花子	333333
4	山本百合子	444444

戻る

なんと、全員のパスワードが表示されてしまった!

MySQL のクエリログを見てみると、次のような SQL が動作したのだとわかる。

```
SELECT * FROM members WHERE id = '' or 1 = 1; -- '
```

この SQL 文を見ると、WHERE 句の id = '' が動作したのではなくて、OR 句の 1 = 1 が動作し、それが条件としてデータを表示したのだということがわかる。

MySQL でクエリログを有効にする手順

MySQL ではクエリログは次の手順で有効にできる。

MariaDB [(none)]> SET GLOBAL general_log_file = 'C:/xampp/mysql/log/mysql.log';

MariaDB [(none)]> SET GLOBAL general_log = 1;

この場合、"C:¥xampp¥mysql" に log フォルダを作成しておく必要がある。mysql.log ファイルは自動で作成される。

MySQL を再起動すると、もとにもどる。

2.2 なぜそうなったのか?

その原因は、プログラムコードに問題がある。

```
$sql = "SELECT * FROM members WHERE id = '{$id}'";
```

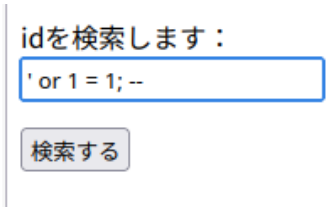
文字列の中に 変数$id を埋めこんだのがいけないのである。そのことによって、悪意のあるユーザーに SQL コマンドの実行を許してしまったのである。

3 対策済みのプログラムコードを実行する

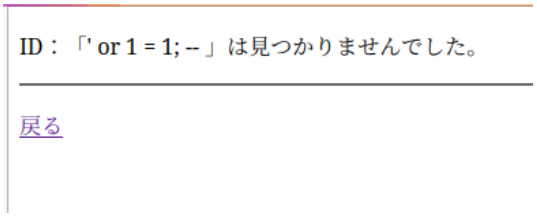
3.1 実行の手順

"work¥after¥" フォルダをドキュメントルートとしてサーバーを起動する。index.php と search.php が動作する。

index.php にアクセスすると、以下のように表示されるので、さっきと同じように `' or 1 = 1; --` と入力する。



今度は、次のように、入力した文字がそのまま表示される。



今回は、次の SQL が動作した。

```
SELECT * FROM members WHERE id = '\' or 1 = 1; -- '
```

"\'" というのは、'(シングルクォーテーション) をエスケープしている。つまり、"' or 1 = 1; -- " が 1 つの文字列として扱われ、その文字列に合致するものを調べるという SQL が動作したのである。

4　プリペアードステートメントの重要性

送信されてきたデータを変数に格納し、それを使って SQL 文を発行するときは、SQL 文の中に変数を埋め込むと、"SQL インジェクション" という攻撃を受ける危険性がある。これは、悪意のあるユーザーに任意を SQL 文の実行を許してしまう。

その対策として、変数を使うときは、SQL 文にプレースホルダーを記述し、変数値をデータとして別個に与えるようにする。そのことにより、ユーザーの入力が SQL 構文として解釈されることがない。

PDO::PARAM_INT って何?

$stm->bindValue(':id', $id, PDO::PARAM_INT) という文で、
PDO::PARAM_INT というのがあるが、
$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false); の場合、
PDO::PARAM_INT は動作していない。
与えられたデータが文字列なら文字列のまま、 数値なら数値のまま、MySQL に渡す。
$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, true); の場合は、
PDO::PARAM_INT は動作する。
与えられたデータを整数に変換して MySQL に渡す。小数点値なら、小数点以下を切り捨てて MySQL に渡す。