بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتا بیس ایران

مدرس: مهندس افشین رفوآ

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 - واحد H 7
88146323 - 88446780 - 88146330

تقدیم به نائب امام عصر آیت الله خامنه ای

که عصا زدنش ضرب آهنگ حیدری دارد.

تقدیم به همه جویندگان علم که توان و امکان شرکت در کلاس های حضوری ما را ندارند.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 - واحد H 7
88146323 - 88446780 - 88146330

برای دریافت فایل Script (اسکریپت) Northwind اینجا را کلیک کنید.

بخش دهم آموزش sql server

تزریق کد به sql server

SQL Injection می تواند پایگاه داده شما را نابود کند.

استفاده از SQL در صفحات وب

در فصل های قبل بیرون کشیدن و بروزرسانی داده های پایگاه داده را با استفاده از SQL آموختید.

وقتی از SQL برای نمایش داده در صفحه وب استفاده می شود معمولا به کاربران وب این اجازه داده می شود تا مقادیر مورد نظرشان جهت جستجو را وارد نمایند.

از آنجا که دستورات SQL تنها متنی هستند، با استفاده از کدهای کامپیوتری می توان براحتی دستورات SQL را به صورت داینامیک تغییر داد تا اطلاعات منتخب را برای کاربر فراهم نماید.

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

مثال بالا یک دستور SELECT در sql server ایجاد کرده و همچنین متغیر txtUserID (تزریق)را به زنجیره SELECT اضافه می نماید. متغیر براساس اطلاعات وارد شده توسط کاربر (Request) از پایگاه داده گرفته شده و در صفحه وب نمایش داده می شود.

ادامه این فصل به خطرات احتمالی ناشی از بکارگیری اطلاعات ورودی کاربران در دستورات SQL می پردازد.

SQL Injection

تکنیکی است که هکرها از آن استفاده کرده تا از طریق ورودی های صفحه وب دستورات SQL را در عبارات SQL تزریق کنند. دستورات تزریق شده می تواند عبارات SQL را تغییر داده و امنیت web application را به خطر اندازد.

بر این اساس که 1=1 همیشه برقرار است. یکبار دیگر به مثال بالا توجه کنید.

فرض کنید که هدف اصلی کد بالا ایجاد عبارت SQL ای است که یک کاربر را با user id مشخصی انتخاب می نماید. اگر راهی نباشد که کاربر را از وارد کردن ورودی "غلط" بر حذر دارد، کاربر می تواند ورودی های "هوشمندی" مانند آنچه در زیر می بینید را وارد نماید.

UserId

1105 or 1=

```
SELECT * FROM Users WHERE UserId = 105 or 1=1
```

عبارت SQL بالا یک عبارت معتبر می باشد. عبارت فوق با تکیه بر این اصل که WHERE 1=1 همیشه صادق است تمام ردیف ها را از جدول Users نشان خواهد داد.

آیا مثال بالا خطرناک به نظر می رسد؟ اگر جدول Users شامل اسم ها و پسورد ها باشد چطور؟

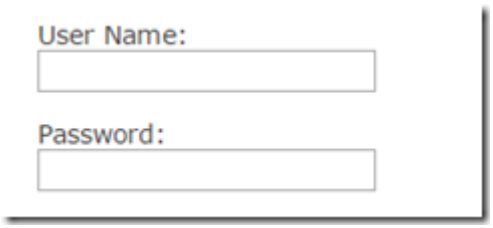
عبارت SQL بالا بسیار شبیه به عبارت زیر می باشد.

```
SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1
```

یک هکر هوشمند میتواند تنها با وارد کردن 105 یا 1=1 در داخل input box به راحتی به تمام اسامی و پسورد های کاربران موجود در یک پایگاه داده دسترسی پیدا کند.

تزریق عبارت در SQL بر این اساس که ""="" همیشه برقرار است

عبارت زیر ساختار رایج مورد استفاده برای شناسایی کاربر جهت ورود او به حساب کاربری خود در یک وب سایت می باشد.




```
uName = getRequestString("UserName");
uPass = getRequestString("UserPass");
sql = "SELECT * FROM Users WHERE Name ='" + uName + "' AND Pass ='" + uPass +"'"
```

یک هکر هوشمند می تواند به راحتی با وارد کردن "" یا "=" در داخل کادر نام کاربردی و پسورد به تمام نام های کاربری و پسورد های کاربران موجود در یک پایگاه داده دسترسی یابد.

این کد در سرور عبارت SQL معتبری مانند شکل زیر ایجاد می کند.

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

SQL result بوجود آمده معتبر بوده و از آنجاکه WHERE """=" همیشه صادق است تمام سطر های جدول Users را نمایش خواهد داد.

تزریق عبارت در SQL بر اساس BATCHED SQL STATEMENTS

اغلب پایگاههای داده ها از BATCHED SQL STATEMENT هایی که با ";" از هم جدا شده اند پشتیبانی می کنند.

```
SELECT * FROM Users; DROP TABLE Suppliers
```

عبارت SQL بالا تمام سطر های جدول Users را نشان داده و جدول Suppliers را حذف خواهد کرد.

اگر سرور کد زیر را داشتیم.

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

به همراه ورودی زیر.

```
UserId
105; DROP TABLE Suppliers
```

کد فوق در سرور عبارت SQL معتبری همانند آنچه در شکل زیر می بینید ایجاد می کرد.

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers
```

پارامترهایی برای محافظت و امنیت از sql server

برخی از توسعه دهندگان وب برای جلوگیری از حملات Injection تعدادی از کلمات و کاراکتر هایی را که می توان در ورودی SQL جهت جستجو وارد کرد در "blacklist"قرار می دهند.

البته این کار ایده خوبی نیست. بسیاری از این کلمه ها (مانند like یا drop و کاراکترها (مانند ; و :) در زبان رایج مورد استفاده قرار می گیرند و باید اجازه استفاده از آنها در اغلب ورودی ها داده شود

(در حقیقت وارد کردن یک عبارت SQL در یک فیلد پایگاه داده ای باید امری کاملا قانونی باشد.)

تنها راه ثابت شده برای محافظت از یک وب سایت در برابر حملات Injection استفاده از پارامتر های SQL است.

پارمتر های SQL مقادیری هستند که در زمان اجرای یک query به شکل کنترل شده ای به آن اضافه می شوند.

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = @0";
db.Execute(txtSQL,txtUserId);
```

توجه داشته باشید که این پارامتر ها در عبارات SQL با نشانگر @ نشان داده می شوند.

موتور SQL هر پارامتر را بررسی می کند تا مطمئن شود که واقعا در ستون درست خود و نه در قسمت اجرایی SQL قرار گرفته است.

```
txtNam = getRequestString("CustomerName");
txtAdd = getRequestString("Address");
txtCit = getRequestString("City");
txtSQL = "INSERT INTO Customers (CustomerName,Address,City) Values(@0,@1,@2)";
db.Execute(txtSQL,txtNam,txtAdd,txtCit);
```

هم اکنون فراگرفتید که چگونه از حملات SQL Injection، یکی از آسیب پذیرترین نقاط هر وب سایت جلوگیری کنید.

مثال ها: مثال های زیر چگونگی ساخت query های پارامتریزه شده را در تعدادی از زبان های رایج وب نشان می دهد.

ASP.NET SELECT

```
txtUserId = getRequestString("UserId");
sql = "SELECT * FROM Customers WHERE CustomerId = @0";
command = new SqlCommand(sql);
command.Parameters.AddWithValue("@0",txtUserID);
command.ExecuteReader();
```

ASP.NET INSERT INTO

```
txtNam = getRequestString("CustomerName");
txtAdd = getRequestString("Address");
txtCit = getRequestString("City");
txtSQL = "INSERT INTO Customers (CustomerName,Address,City) Values(@0,@1,@2)";
command = new SqlCommand(txtSQL);
command.Parameters.AddWithValue("@0",txtNam);
```

```
command.Parameters.AddWithValue("@1",txtAdd);

command.Parameters.AddWithValue("@2",txtCit);

command.ExecuteNonQuery();
```

PHP INSERT INTO

```
$stmt = $dbh->prepare("INSERT INTO Customers (CustomerName,Address,City)

VALUES (:nam, :add, :cit)");

$stmt->bindParam(':nam', $txtNam);

$stmt->bindParam(':add', $txtAdd);

$stmt->bindParam(':cit', $txtCit);

$stmt->execute();
```

در بخش بعدی از سری مقالات آموزشی sql server با نحوه استفاده از دستور SELECT TOP در sql server آشنا می شویم.