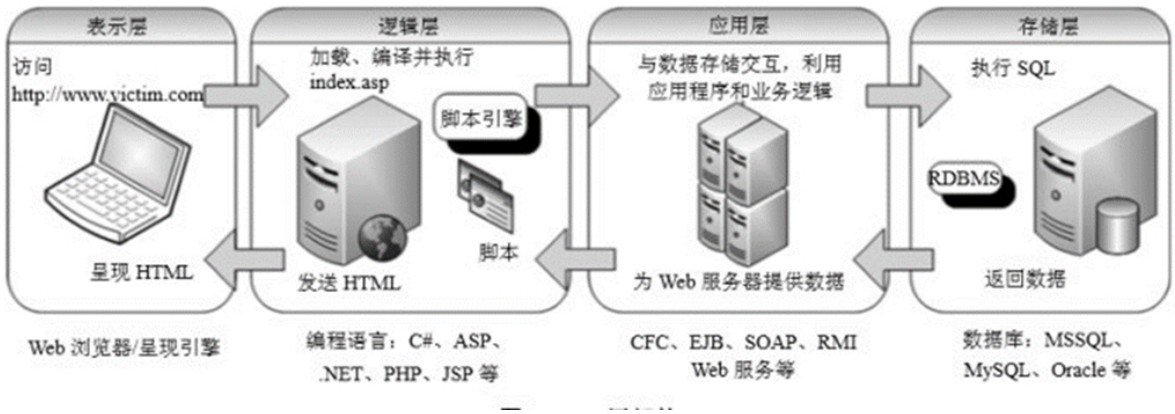
web安全之SQL注入基础

SQL注入基础

SQL注入简介

Web请求响应过程



什么是SQL注入?

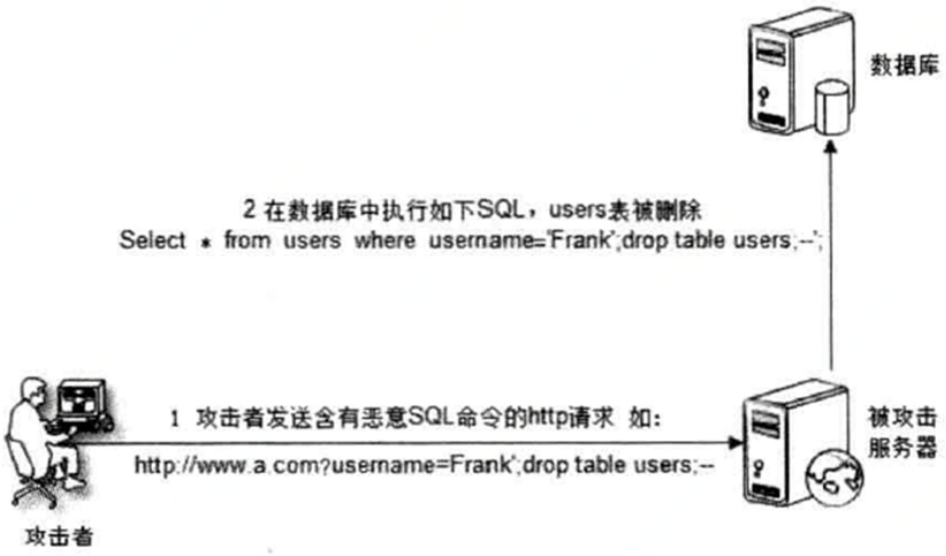
就是指web应用程序对用户输入数据的合法性没有判断，前端传入后端的参数是攻击者可控的，并且参数带入数据库查询，攻击者可以通过构造不同的SQL语句来实现对数据库的任意操作。

为什么会产生sql注入?

开发人员可以使用动态SQL语句创建通用，灵活的应用。动态SQL语句是在执行过程中构造的，它根据不同的条件产生不同的sql语句。当开发人员在运行过程中需要根据不同的查询标准决定提取什么字段（如select语句）,或者根据不同的条件选择不同的查询表时，动态地构造SQL语句会非常有用。

> Php语句为列:
>
> `$query="SELECT * FROM users WHERE id = $_GET['id']";`

由于这里的参数ID可控，且带入数据库查询，所以非法用户可以任意拼接SQL语句进行攻击。



SQL注入的原理

Sql注入漏洞的产生需要满足那两个条件?

参数用户可控：前端传给后端的参数内容是用户可以控制的。

参数带入数据库查询：传入的参数拼接到sql语句，且带入数据库查询。

当传入ID参数为1'时，数据库执行的代码如下所示。

```
sehlect * from users where id =1'
```

这样是会报出错误的，因为这不符合数据库语法规范。

当传入的ID参数为and 1=1 时，执行的SQL语句如下所示。

```
select * from users where id = 1' and 1=1
```

因为1=1为真，且where语句中id 1=1也为真，所以页面会返回与id=1相同的结果。

当传入的ID参数为and 1=2时，由于1=2不成立，所以返回假，页面就会返回与id=1不同的结果。

由此可以初步判断ID参数存在SQL注入漏洞，攻击者可以进一步拼接SQL语句进行攻击，致使数据库信息泄露，甚至进一步获取服务器权限等。

——

在实际环境中，凡是满足上述两个条件的参数皆可能存在SQL注入漏洞，因此开发者需秉持"外部参数皆不可信的原则"进行开发。

——

MySQL注入相关的知识点

Mysql数据库结构

数据库A=网站A

表名

列名

数据

数据库 B=网站B

```
数据库
```

Mysql数据库自带信息

在MySql5.0 版本之后，MySql默认在数据库中存放一个“information_schema”的数据库，在该库中，读者需要记住三个表名，分别是

`SCHEMATA,TABLES,COLUMNS`

`SCHEMNSz表存储该用户创建的所有数据库的库名，我们需要记住该表中记录数据库库名的字段名为SCHEMA_NAME`

`TABLES`表存储该用户创建的所有数据库的库名和表名，我们需要记住该表中记录数据库库名和表名的字段名分别为`TABLE_SCHEMA和TABLE_NAME`。

`COLUMNS`表名和字段名的字段名为`TABLE_SCHEMA，TABLE_NAME和COLUMN_NAME`。

Mysql查询语句

在不知道任何条件时，语句如下所示。

`SELECT 要查询的字段名 FROM 库名.表名`

在知道一条已知条件时，语句如下所示。

`SELECT 要查询的字段名 FROM 库名.表名 WHERE 已知条件的字段名=‘已知条件的值’`

在知道两条已知条件时，语句如下所示。

`SELECT 要查询的字段名 FROM 库名.表名 WHERE 已知道条件1的字段名=‘已知条件1的值’ AND 已知条件2的字段名=‘已知条件2的值’`

Limit 的用法

Limit的使用格式为limit m,n, 其中m是指记录开始的位置，从0开始，表示第一条记录;n是指取n条记录。

例如limit 0,1表示从第一条记录开始，取一条记录，

需要记住的几个函数

`Database():当前网站使用的数据库`

`Version():当前MySQL的版本`

`User()`:当前MySQL的用户

注释符号

在MySQL中，常见注释符的表达方式：`#`或`-- 空格`或`/**/`

内联注释

内联注释的形式：*/! code/*。内联注释可以用于整个SQL语句中，用来执行我们的SQL语句，下面举一个列：

```
index.php?id=-15 /*!UNION*/ /*!SELECT*/ 1,2,3
```