脚本 PHP 的 SQL 注入防护实验

一【实验目标】

- 了解如何防护 SQL 注入
- 了解如何对 PHP 代码修改进行 SQL 注入防护

二【实验环境】

- Windows 10
- XAMPP

三【实验原理】

脚本 PHP 的 SQL 注入防护实验原理如下：

SQL 注入被那些菜鸟级别的所谓黑客高手玩出了滋味，发现现在大部分黑客入侵都是基于 SQL 注入实现的。现在我开始说如果编写通用的 SQL 防注入程序一般的 http 请求不外乎 get 和 post。所以只要我们在文件中过滤所有 post 或者 get 请求中的参数信息中非法字符即可，所以我们实现 http 请求信息过滤就可以判断是是否受到 SQL 注入攻击。当传递数据后，PHP 解析器会分析数据的信息，然后根据"&"，分出各个数组内的数据所以 get 的拦截如下：

首先我们定义请求中不能包含如下字符,并进行替换保存到一个 phpsqlinj.php 文件中,代码如下：

```
<?php
function php_sava($str)
{
$farr = array(
  "/s /",
"/<(/?)(script|i?frame|style|html|body|title|link|meta|?|%)([^>]*?)>/isU",
"/(<[^>]*)on[a-zA-Z] s*=([^>]                                    *>)/isU",     );
$tarr = array(
  " ",
"＜\1\2\3＞",      //如果要直接清除不安全的标签，这里可以留空
"\1\2",
    );
```

```
$str = preg_replace( $farr,$tarr,$str);
      return $str;
}
//php sql 防注入代码
class sqlin
{
//dowith_sql($value)
function dowith_sql($str)
{
      $str = str_replace("and","",$str);
      $str = str_replace("execute","",$str);
      $str = str_replace("update","",$str);
      $str = str_replace("count","",$str);
      $str = str_replace("chr","",$str);
      $str = str_replace("mid","",$str);
      $str = str_replace("master","",$str);
      $str = str_replace("truncate","",$str);
      $str = str_replace("char","",$str);
      $str = str_replace("declare","",$str);
      $str = str_replace("select","",$str);
      $str = str_replace("create","",$str);
      $str = str_replace("delete","",$str);
      $str = str_replace("insert","",$str);
      $str = str_replace("'","",$str);
      $str = str_replace(" ","",$str);
      $str = str_replace("or","",$str);
      $str = str_replace("=","",$str);
      $str = str_replace(" ","",$str);
    //echo $str;
    return $str;
```

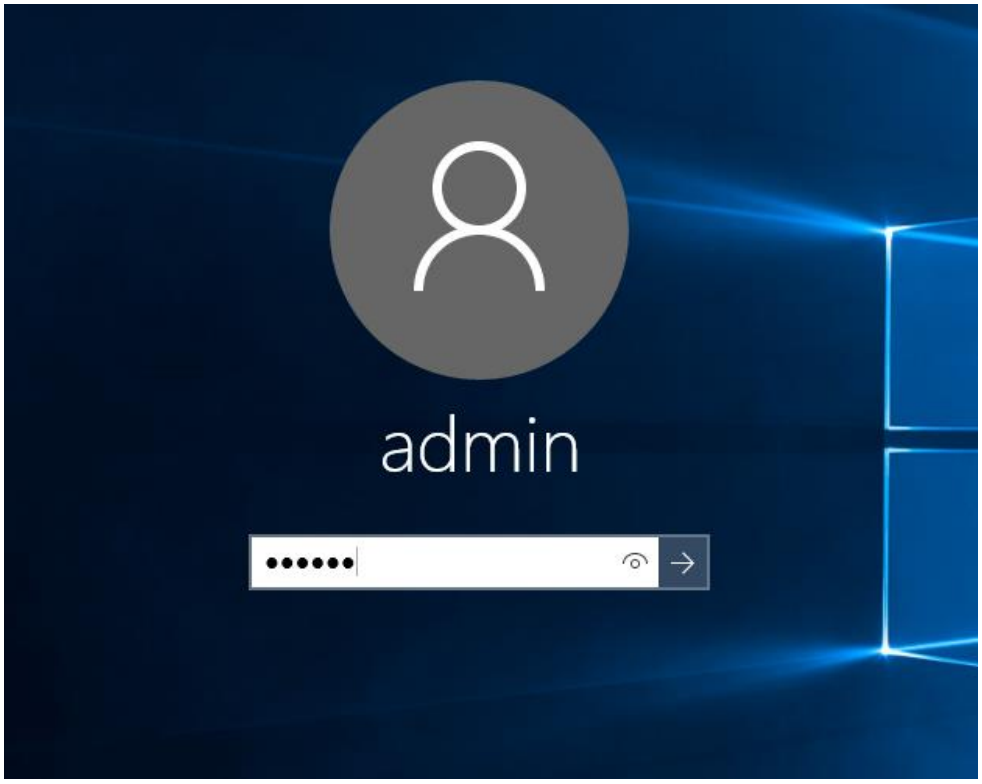
```
}
 //aticle()防 SQL 注入函数//php 教程
 function sqlin()
 {
    foreach ($_GET as $key=>$value)
 {
    $_GET[$key]=$this->dowith_sql($value);
 }
    foreach ($_POST as $key=>$value)
 {
    $_POST[$key]=$this->dowith_sql($value);
 }
}
}
    $dbsql=new sqlin();
    ?>
```

然后在获取 form 表单内容页面,例如 login.php 中加入一行代码 include ('phpsqlinj.php');即可。

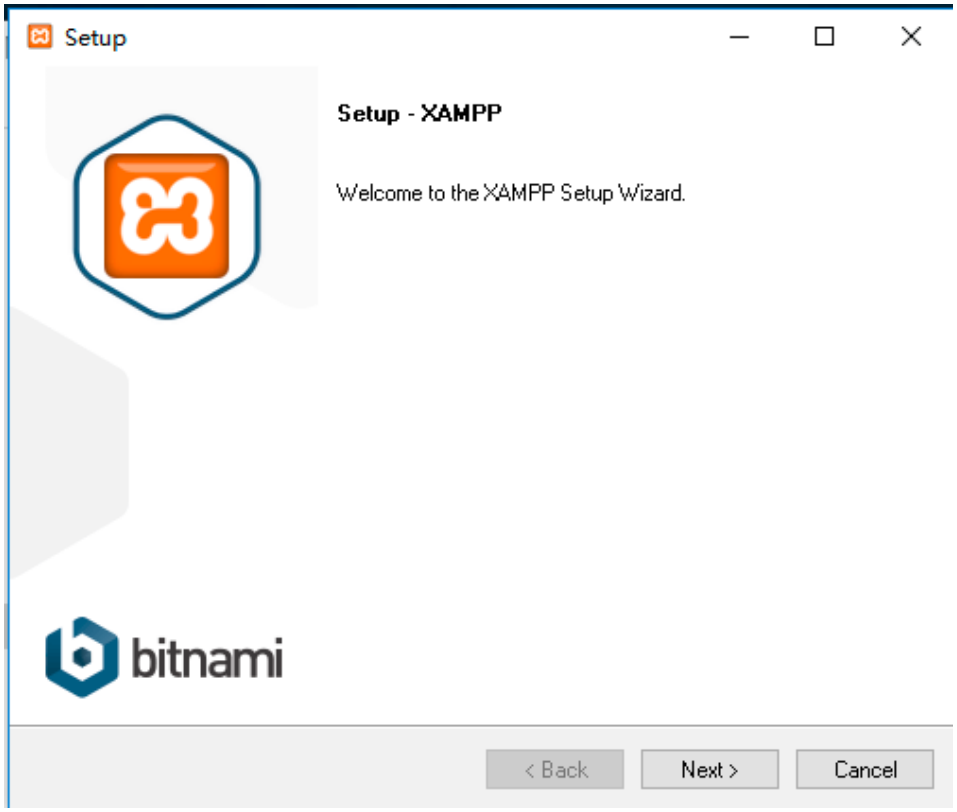
这样就实现了 get 和 post 请求的信息拦截，只需要在 conn.asp 打开数据库文件之前引用这个页面即可不用再考虑是否还会受到 SQL 注入攻击。

四【实验步骤】

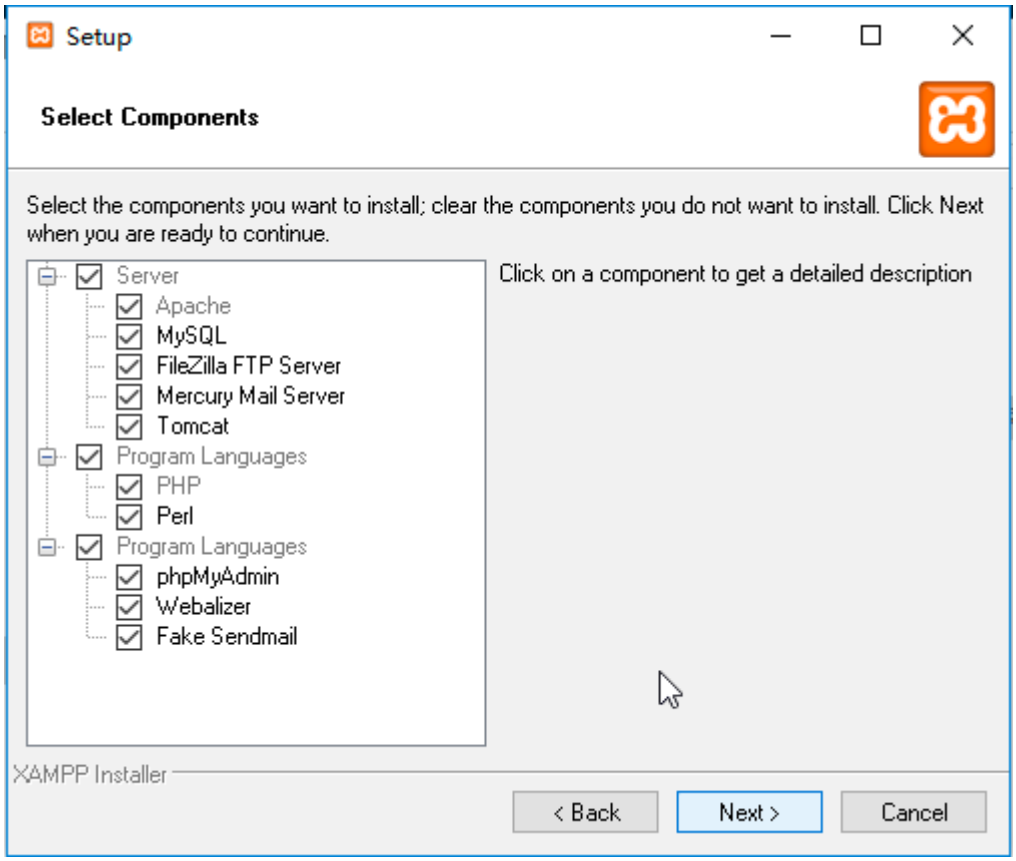
1、输入密码【Admin123456】，登录系统。如下图所示。



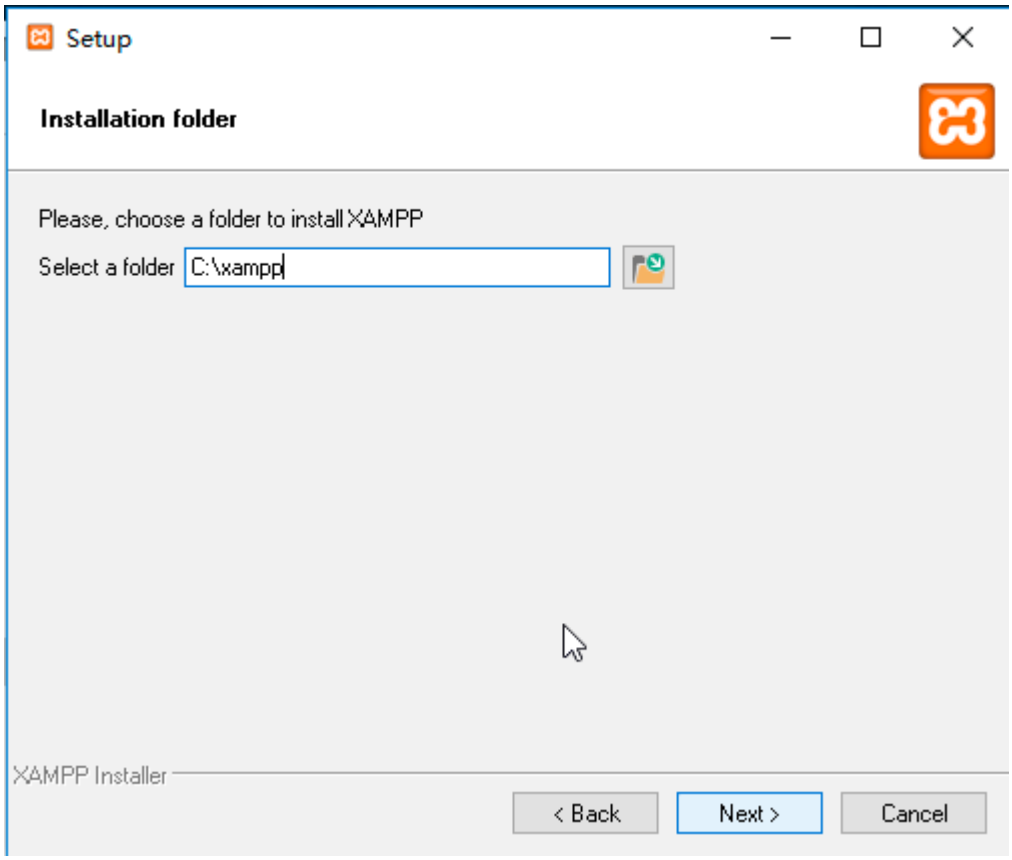
2、双击 C:\tools 中的 XAMPP 安装程序，安装 XAMPP。



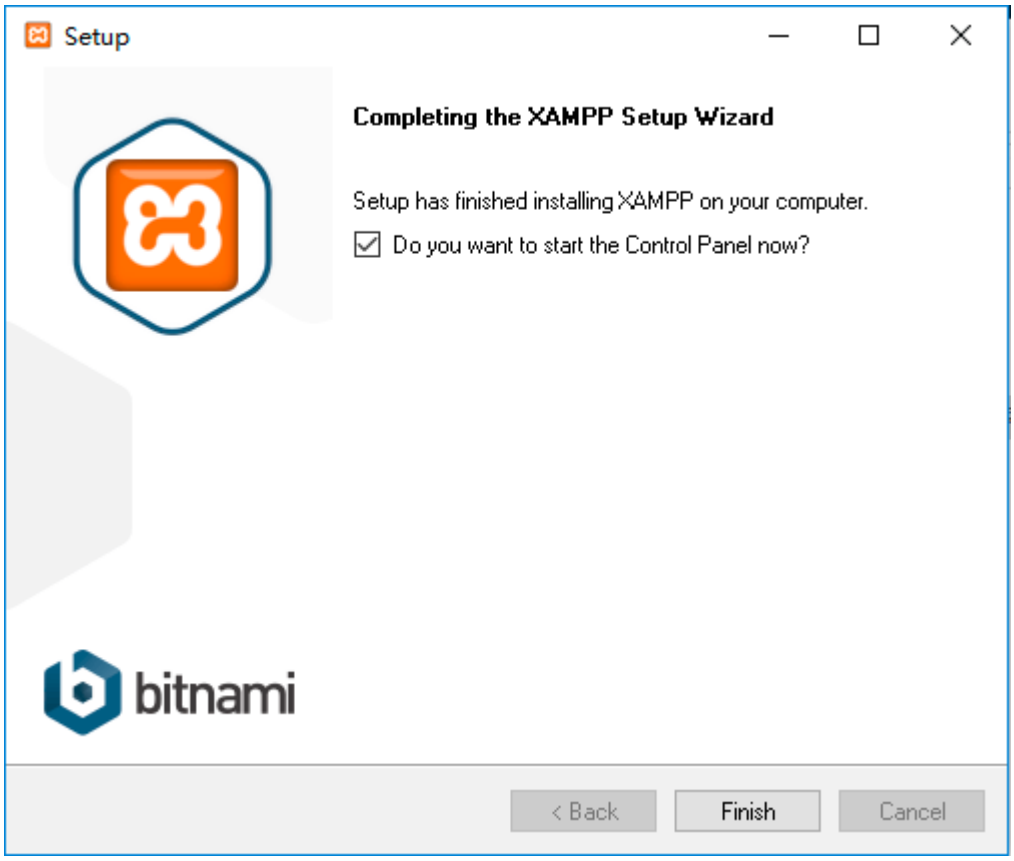
3、点击 next 进行默认安装



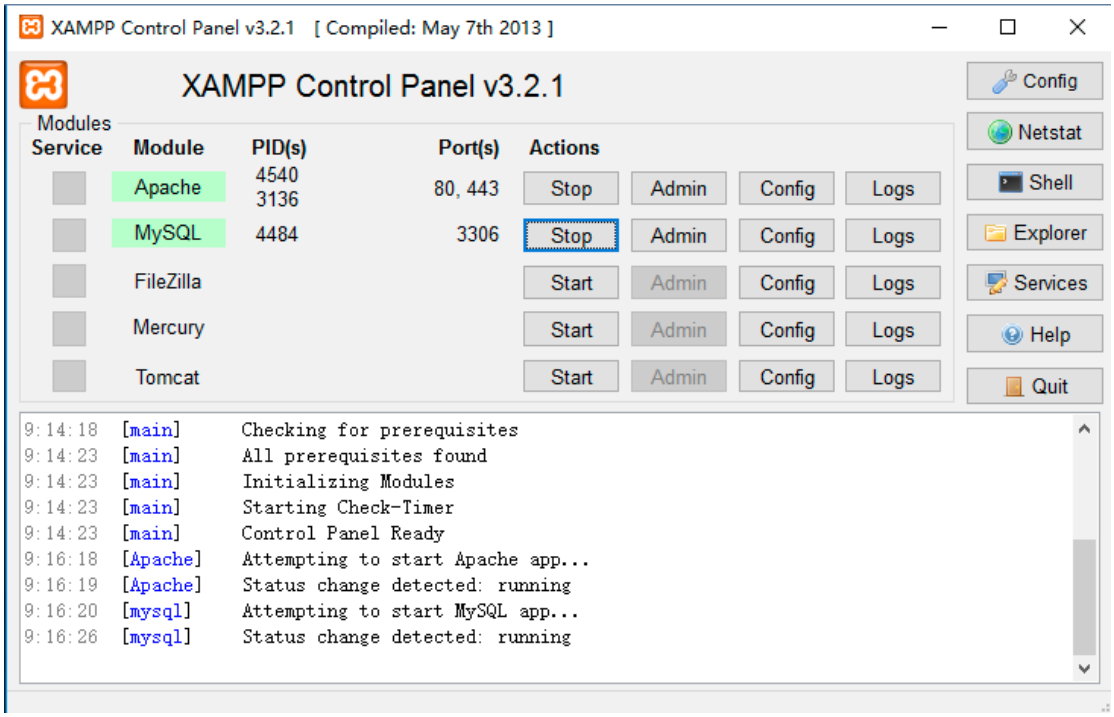
4、选择安装路径，点击 next。



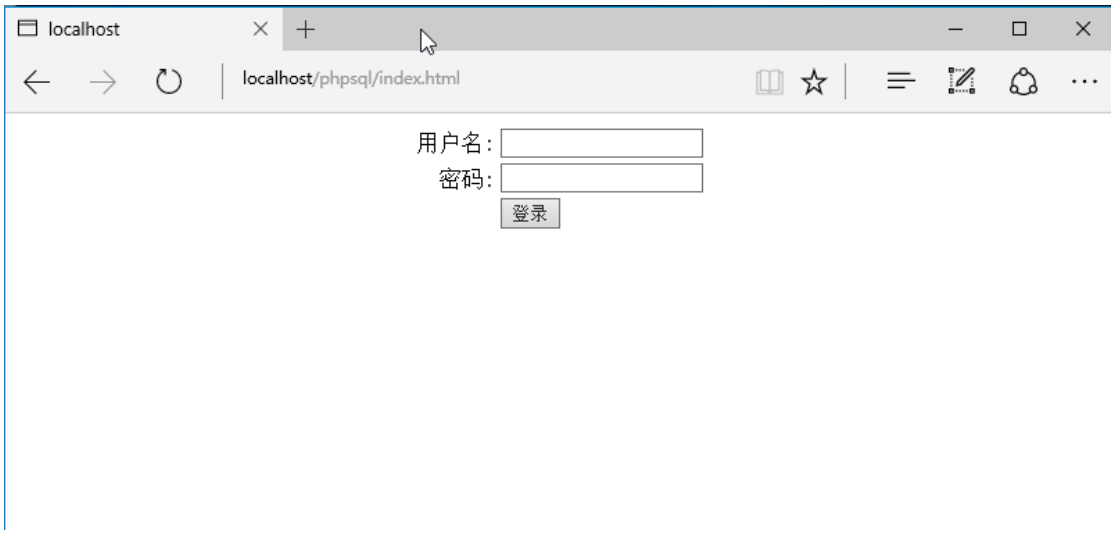
5、安装完成，点击 finish。



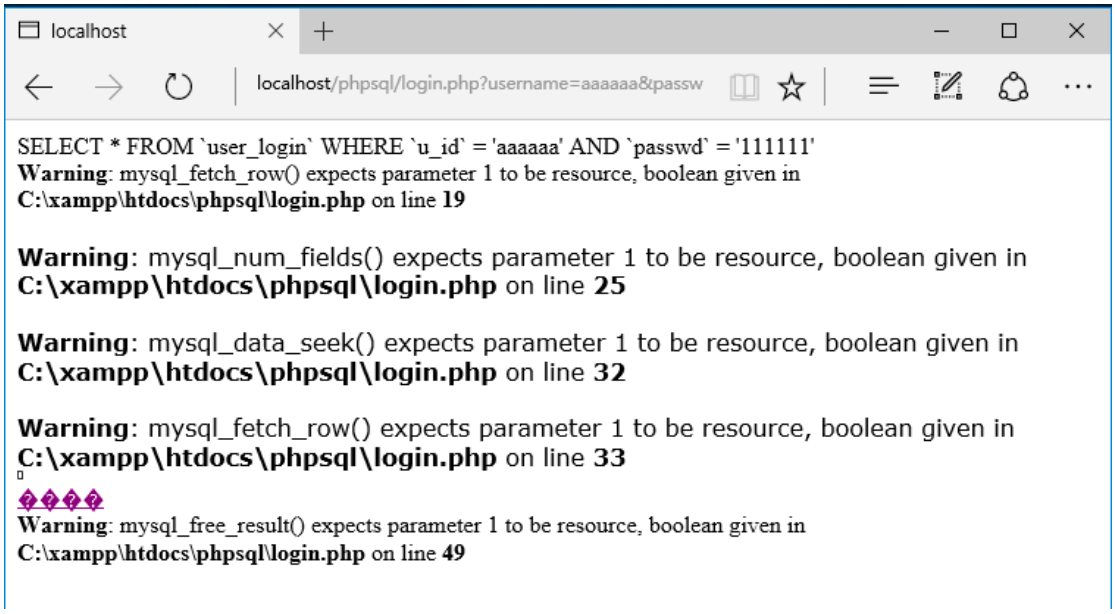
6、将桌面上 tools 文件夹内的 phpsql 文件夹放到 C:\xampp\htdocs 文件夹中，将 lxr 文件夹放到【C:\xampp\mysql\data】中。再打开【C:\xampp\ xampp-control】，然后开启 Apache 和 Mysql 服务，点击 start 出现如下所示便说明成功开启服务。如下图所示。



7、打开 IE 浏览器，输入【http://localhost/phpsql/index.html】。如下图所示。

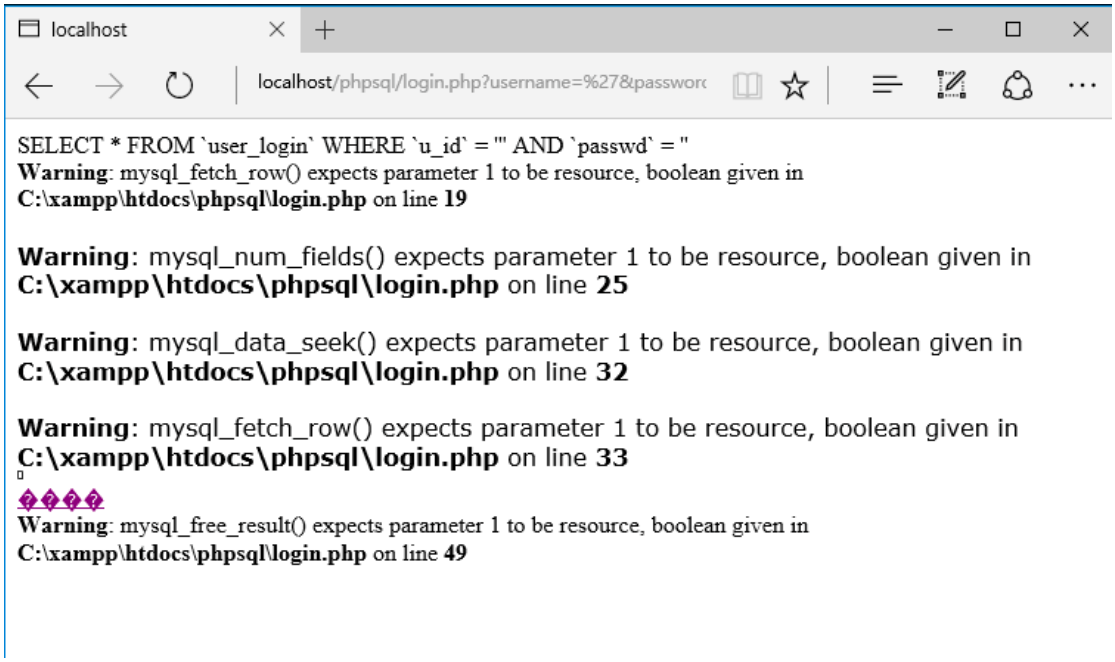


8、假设我们知道用户名为：aaaaaa，密码：111111。我们输入正确的用户名和密码，点击【登录】。如下图所示。

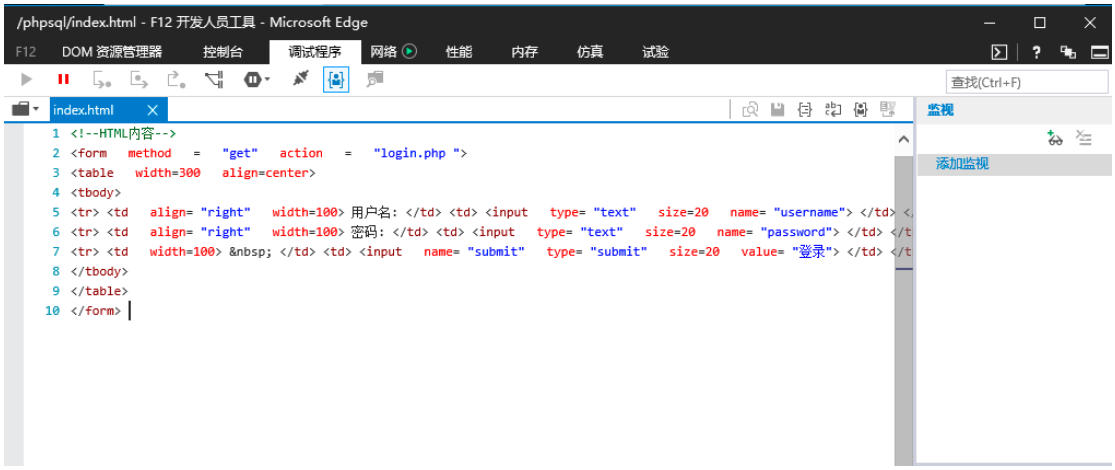


9、在对一个网站进行安全检测的时候，需要对其进行一些手动探测，检测者在【用户】框输入一个单引号，密码留空，点击【登录】。如下图所示。

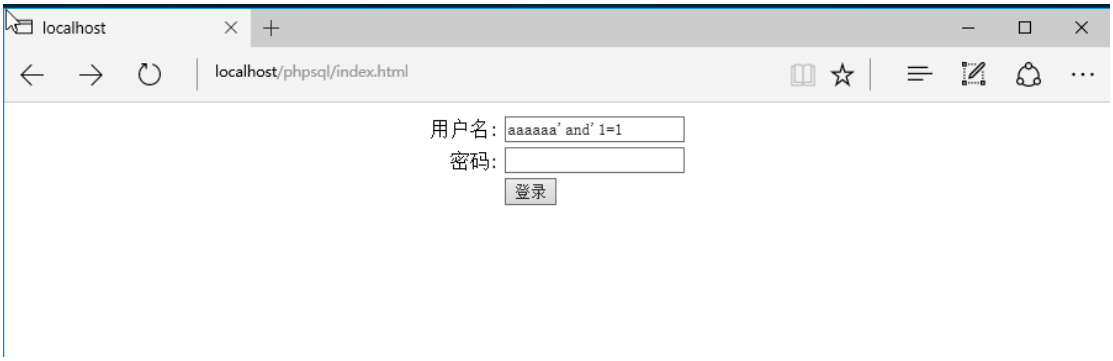
10、返回结果如下图所示。



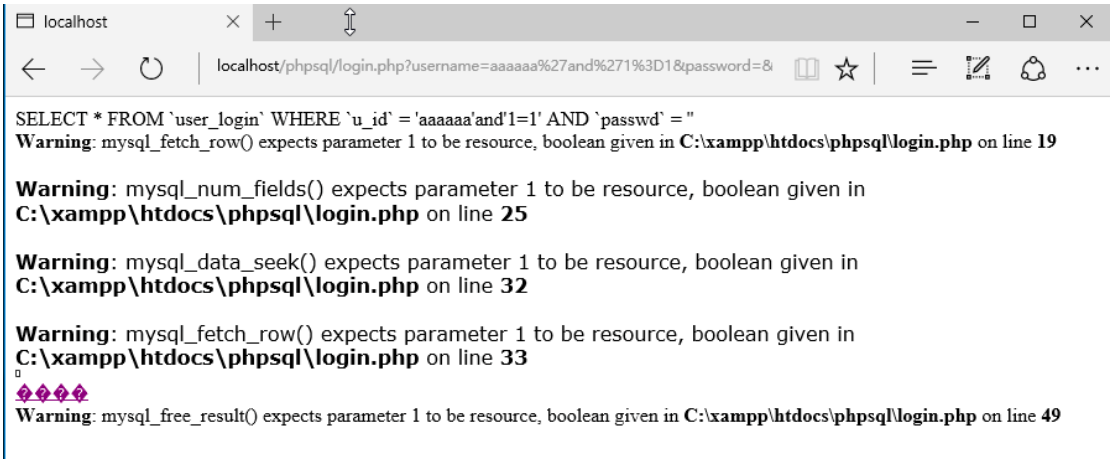
11、从返回信息可以得知系统使用了 PHP+MYSQL 的架构，且很有可能存在含注入漏洞的 SQL 语句，如【SELECT * FROM user_login WHERE u_id='’’】。在登陆界面中右键单击，选择 IE 浏览器的查看 index.html 源文件，读取 html 源码。如下图所示。



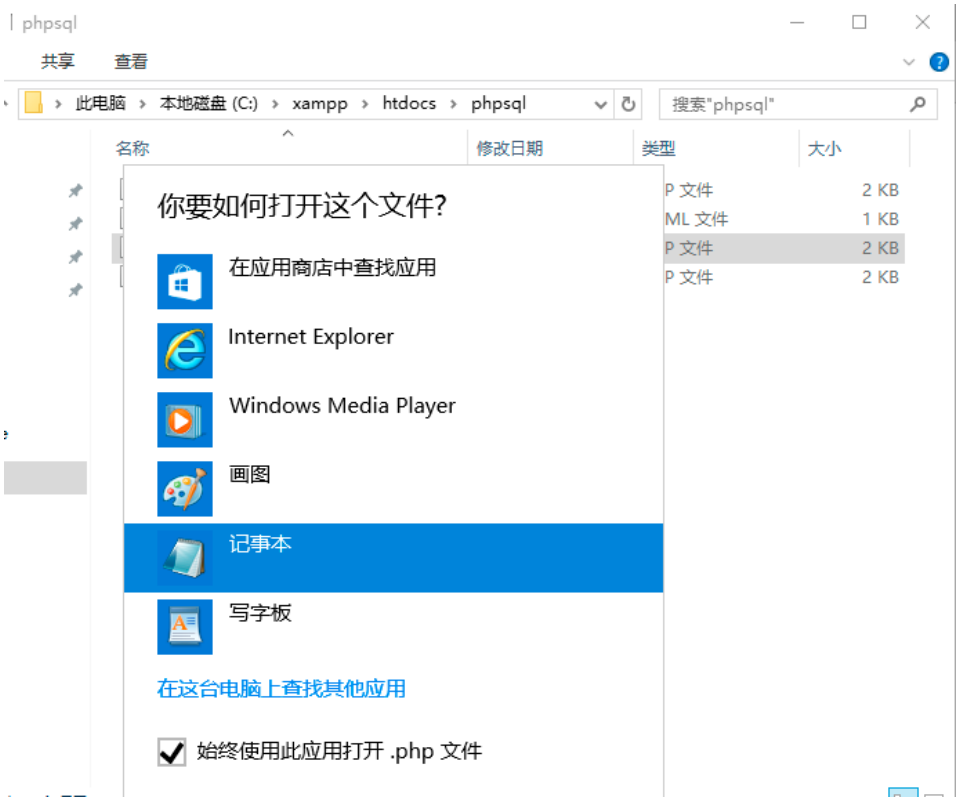
12、通过分析 html 源码，可以知道提交的 username、password 字段、post 提交方式和 login.php 的处理页面，后台的 SQL 语句确定是【 SELECT * FROM user_login WHERE u_id='$username'】，当我们输入用户名为单引号时，造成语法错误。登陆时 username 输入框输入【aaaaaa' or '1=1】即可。如下图所示。



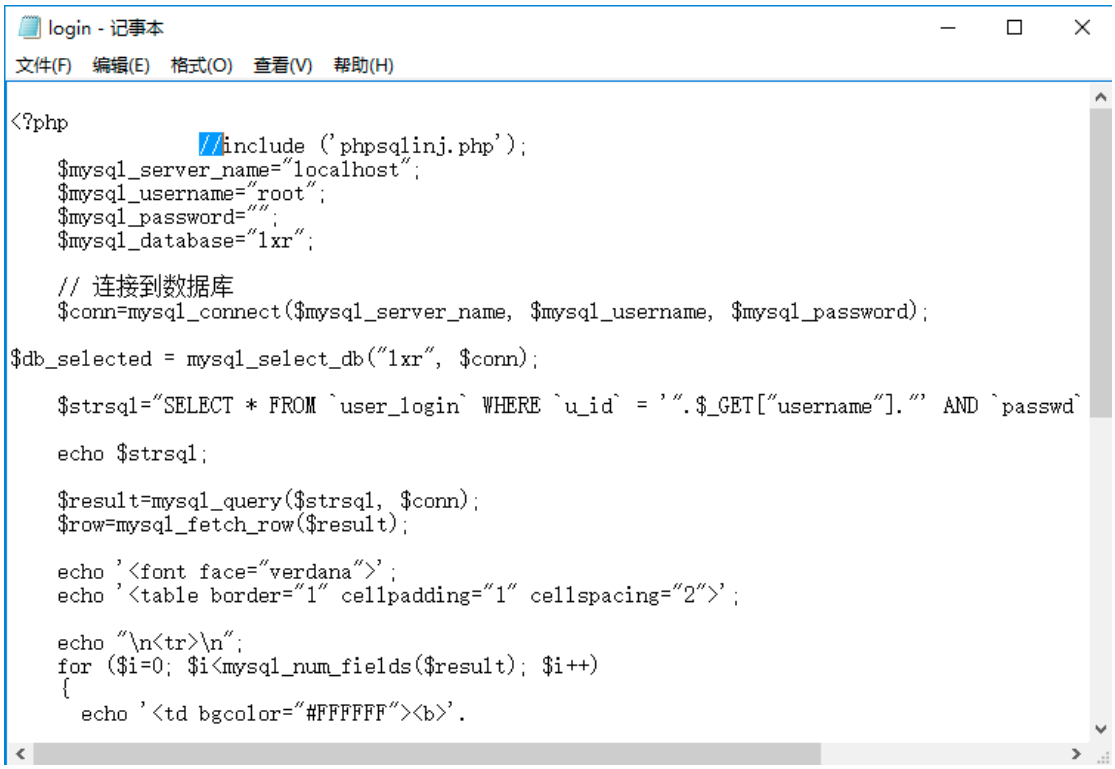
13、【aaaaaa' or '1=1 】构建了 SELECT * FROM 'user_login' WHERE 'u_id' = 'aaaaaa' or '1 = 1' AND 'passwd' = ''语句，直接通过 1=1 的恒等条件，让验证通过。如下图所示。



14、使用记事本打开【c:\xampp\htdocs\phpsql\login.php】文件，点击打开选择打开方式记事本即可。如下图所示。

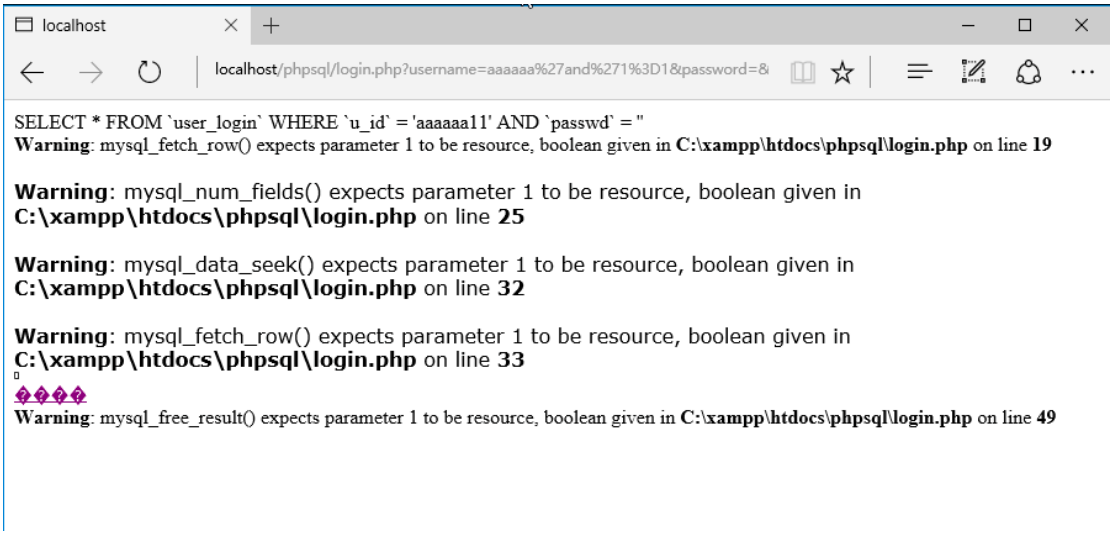


15、将 include ('phpsqlinj.php')的注释去掉，去掉//。如下图所示。



16、输入【http://localhost/phpsql/index.html】，并输入用户名：aaaaaa'or ' 1 = 1 ，点击登陆进行验证。如下图所示。

17、验证发现 u_id = aaaaaa11 并出错，说明 sql 防注入成功。如下图所示。



五【实验思考】

- 是否有其他方法对漏洞进行检测？