北京邮电大学网络空间安全学院
信息安全认知实习
实验报告

课程名称：信息安全认知实习

单元名称：SQL 注入实验

姓名：任子恒

学号：2017522133

班级：2017661801

专业：信息安全

指导教师：颉夏青

成绩：

日　　期：2018 年 9 月 19 日

一、 实验目的

通过一个 SQL 攻击示例，初步了解 SQL 语言及 SQL 注入攻击原理，为后续专业课学习打下基础。

二、 实验原理

2.1 SQL 语言介绍

SQL 是一个结构化的查询语言，用于存取数据以及查询、更新和管理关系数据库系统。

其查询语句为 SELECT <statement> FROM <table> WHERE <condition>

删除语句为 DELETE FROM <table> WHERE <condition>

更新语句为 UPDATE <table> SET <field> = <value> WHERE <condition>

2.2 SQL 注入攻击

网站中涉及到填写表单的部分可能涉及到了 SQL 语句，例如在系统登录部分，可能有类似于 SELECT * From Table WHERE Name='xx' and Password='yy' 的语句，“xx”部分即为登录用户名，通过在用户名框中输入包含可插入上述 SQL 语句的字符串，可以欺骗服务器，绕过密码验证。

例如在用户名框中输入“张三' or 1=1;#”（不含双引号），上述查询语句变为“SELECT * From Table WHERE Name='张三' or 1=1;#' and Password='yy'”（不含双引号），#起到了注释掉后面密码判断语句的作用，1=1 是一个永真句，永远为真，一旦有一个真句，整句为真，所以上述语句的执行结果一定为真，一定可以登录成功。

如果不输入 *or 1=1*，那么上述查询语句变为“SELECT * From Table WHERE Name='张三' ;#' and Password='yy'”，一旦数据库中存在该用户名，执行结果即为真，将可绕过该账号的密码验证阶段。

三、 实验环境

一台装有 IE 11 的 Windows10 电脑。

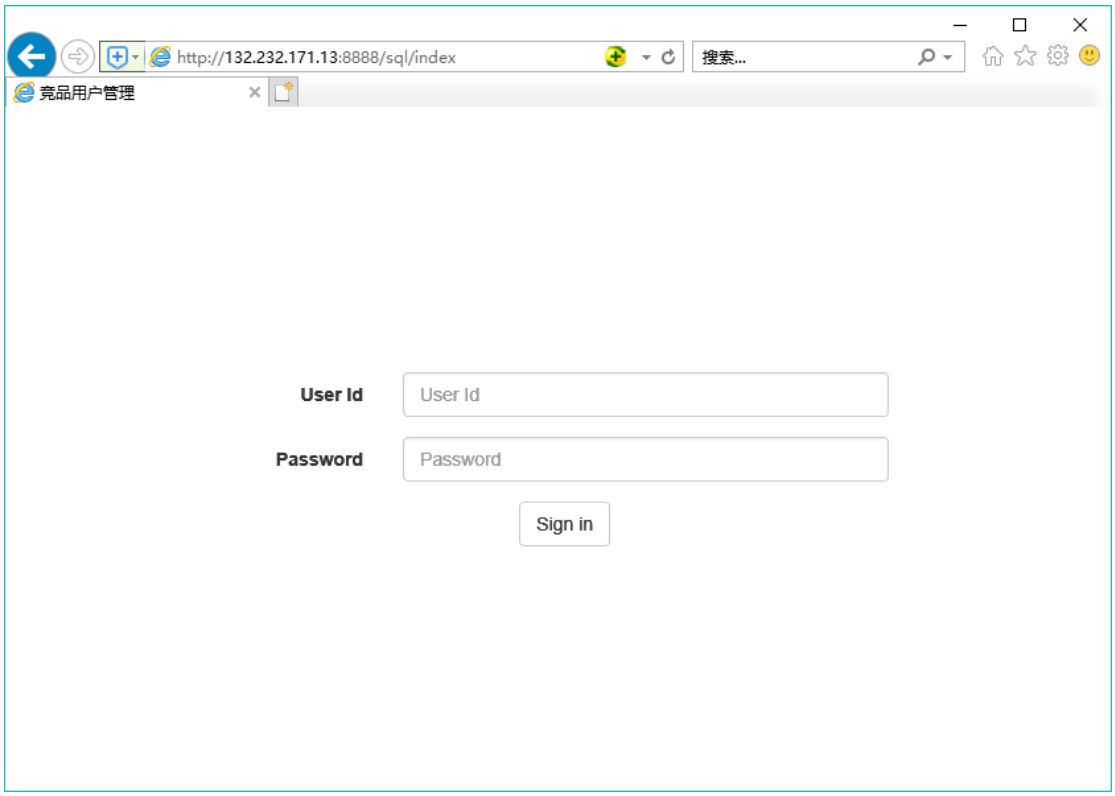
本实验对于操作系统需求不是很大

四、 实验过程及遇到的问题分析

4.1 实验过程

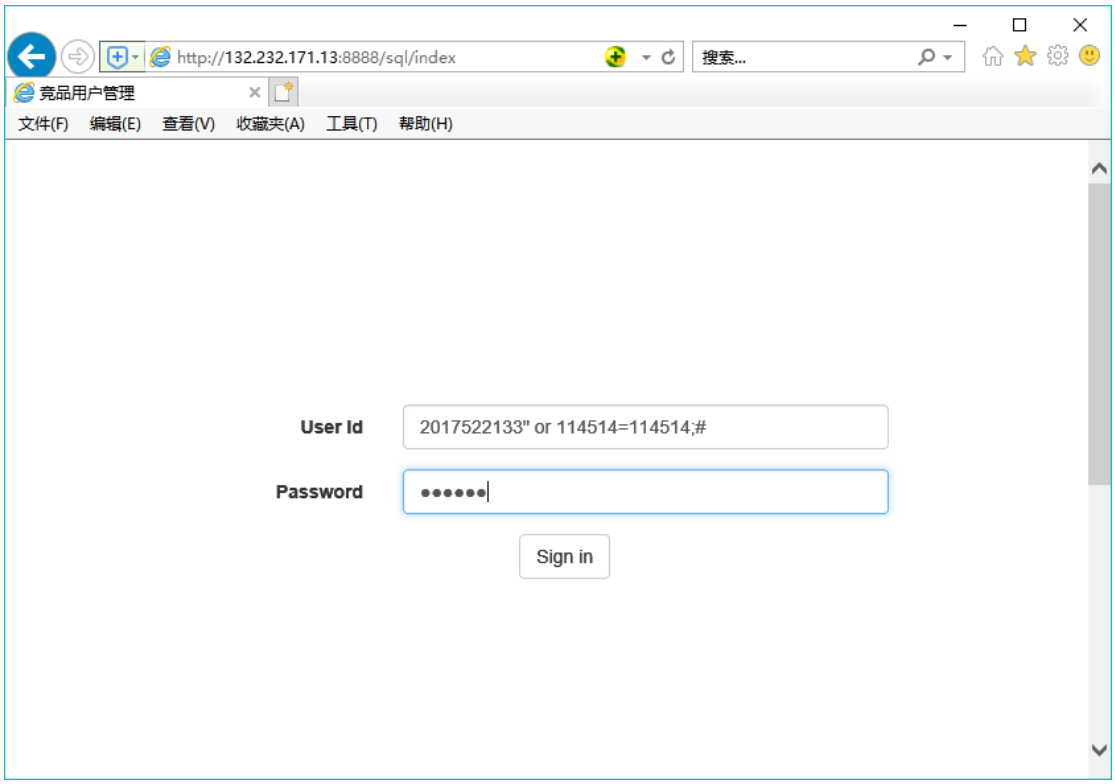
1. 进入一个有 SQL 注入漏洞存在的网站，

本例为 http://132.232.171.13:8888/sql/index

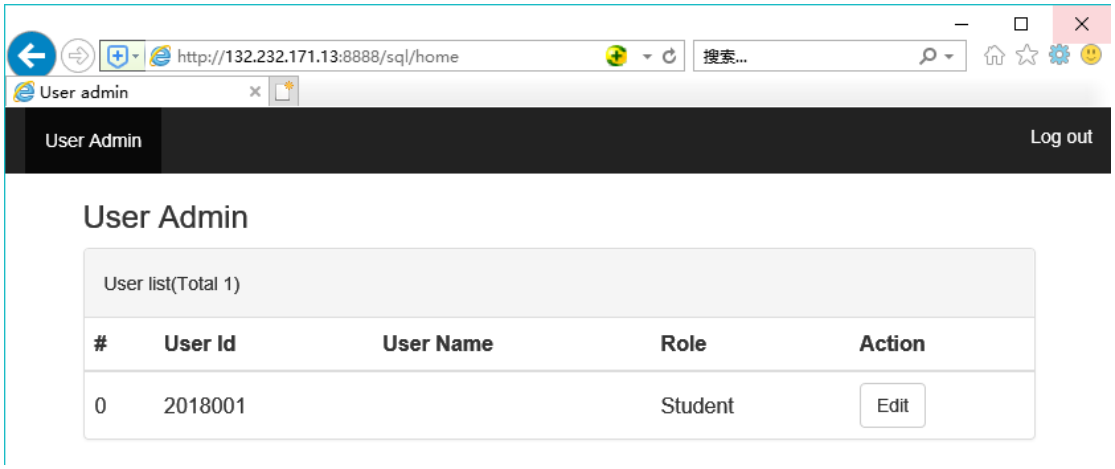


2. 在用户栏中输入 *2017522133” or 114514=114514; #*

密码栏中输入任意内容。

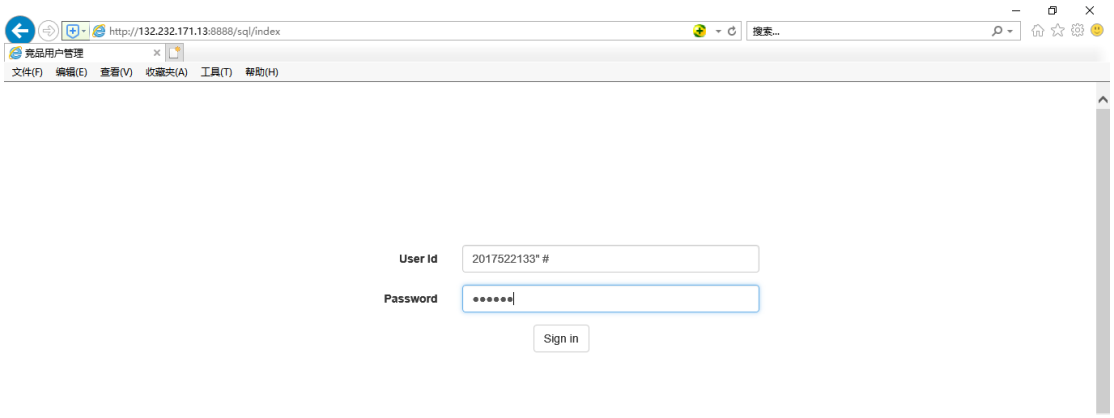


3. 登陆成功，进入系统。但是用户 id 仍然是 2018001，下面我们会让它变成自己的学号。

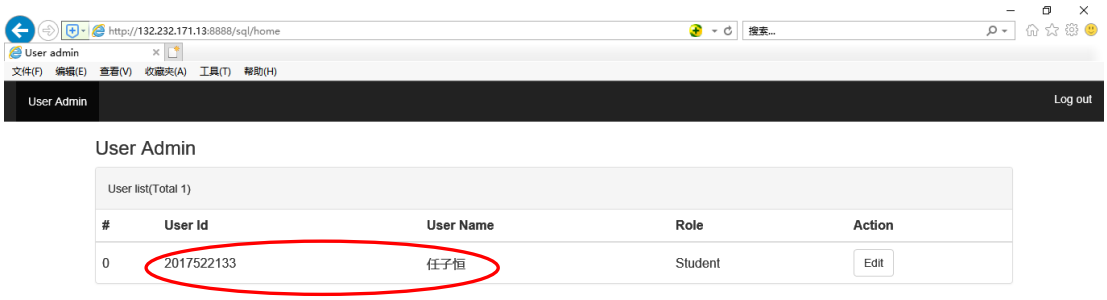


4. 退出登录当前账号，这时在用户名栏中输入 *2017522133”;#*

密码栏中仍输入任意内容。



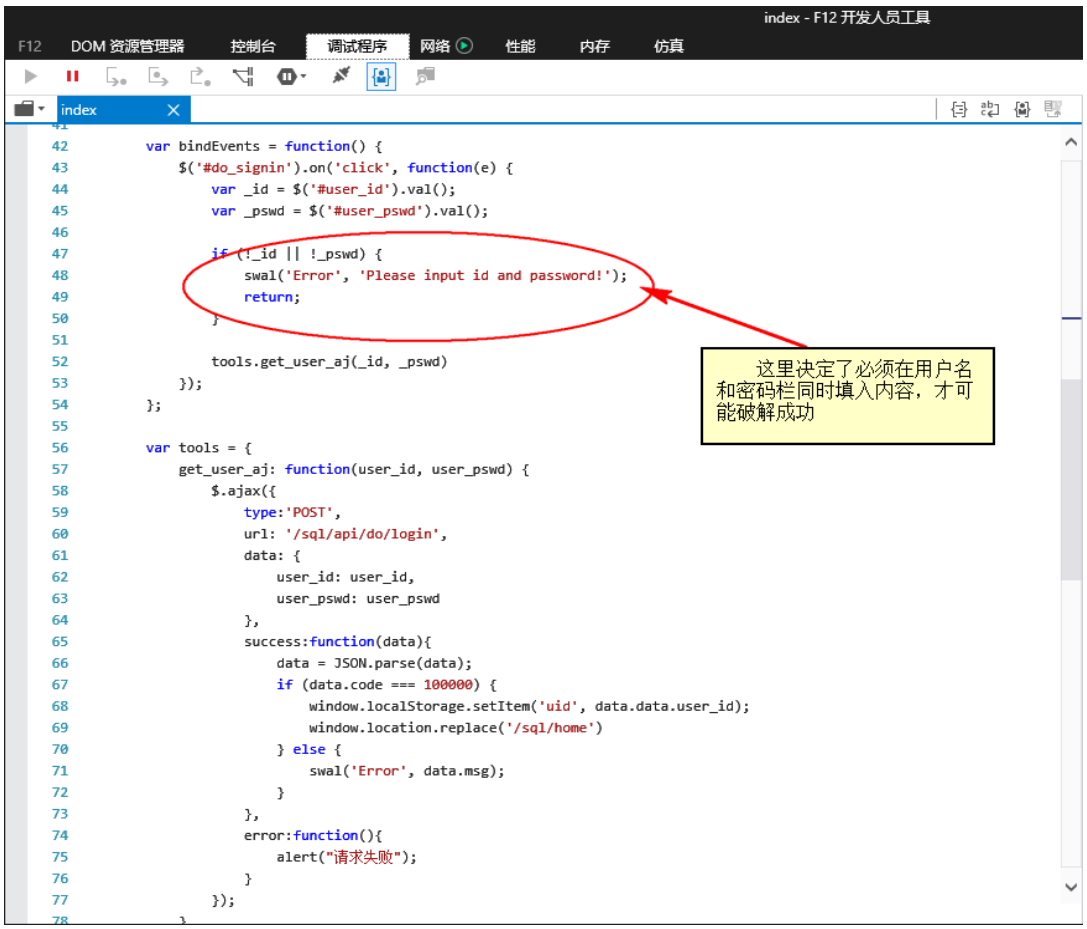
5. 成功！现在 userid 也是自己的学号了。



4.2 问题分析

一、网站管理员应如何防御 SQL 攻击？

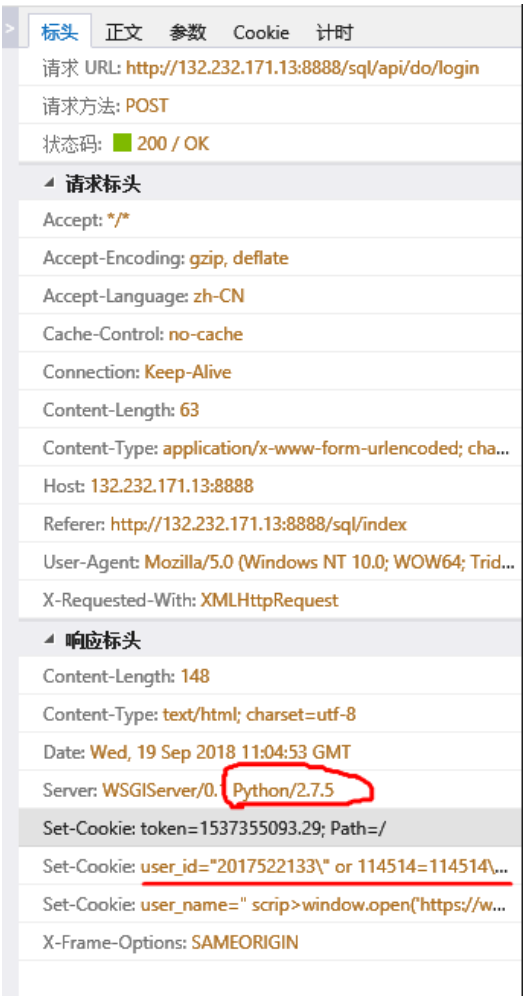
1. 应尽可能隐藏和压缩网页代码，减少让人通过分析代码得出破解方法的可能，本例中，网页的代码也对 SQL 攻击起到了一些辅助的作用。




```
42          var bindEvents = function() {
43              $('#do_signin').on('click', function(e) {
44                  var _id = $('#user_id').val();
45                  var _pswd = $('#user_pswd').val();
46
47                  if (!_id || !_pswd) {
48                      swal('Error', 'Please input id and password!');
49                      return;
50                  }
51
52                  tools.get_user_aj(_id, _pswd)
53              });
54          };
55
56          var tools = {
57              get_user_aj: function(user_id, user_pswd) {
58                  $.ajax({
59                      type:'POST',
60                      url: '/sql/api/do/login',
61                      data: {
62                          user_id: user_id,
63                          user_pswd: user_pswd
64                      },
65                      success:function(data){
66                          data = JSON.parse(data);
67                          if (data.code === 100000) {
68                              window.localStorage.setItem('uid', data.data.user_id);
69                              window.location.replace('/sql/home')
70                          } else {
71                              swal('Error', data.msg);
72                          }
73                      },
74                      error:function(){
75                          alert("请求失败");
76                      }
77                  });
78              }
```

```
74                      error:function(){
75                          alert("请求失败");
76                      }
77                  });
78              }
79          };
80
81          var initPlugins = function() {
82              if(window.localStorage.getItem('uid')) {
83                  window.location.replace('/sql/home');
84              }
85          };
86
87          var main = function() {
88              bindEvents();
89              initPlugins();
90          }
91
92          main();
93      </script>
94  </html>
```

有大量代码带有与 sql 目录有关的操作，甚至可以直接看出上述代码对应的逻辑。



事实上，源代码是用 python 写的。这也是注入时使用“#”的理由。

2. 控制用户输入。拒绝接受带有任何可疑字符的输入，防止其与查询代码发生拼接而造成非法的查询操作。

3. 在需要有查询操作的元素中，不要使用动态 SQL 语句。

二、为什么用户名没有变成自己的学号（不符合预期）？

注入时永真句使用多余，就算查找语句判断为真，也只是绕过了密码验证，未对需要登录的用户名进行赋值，此时需要更换语句。